

Festo PA-Toolkit Bedienungsanleitung

Datum
24.08.2022

Ersteller
BUNK

Version
2.0.5.0

1	Allgemeine Hinweise.....	11
1.1	Versionsverwaltung	11
1.1.1	Änderungshistorie.....	12
1.1.2	Release 1.0.1.1, 1.0.1.2, 1.0.1.3 und 2.0.5.0	13
2	Lizenz für PA Toolkit.....	14
2.1	Kurzbeschreibung	14
2.2	Demolizenz	14
2.3	Lizenz erwerben und installieren.....	15
2.4	Symbol.....	15
2.5	Faceplate.....	16
2.5.1	Main	16
3	Analogwertanzeige	17
3.1	Funktionsbaustein AnaView	17
3.1.1	Kurzbeschreibung	17
3.1.2	Grundfunktionen	17
3.1.3	Funktionen	17
3.1.4	FB-spezifischer Worst Quality Code (WQC)	19
3.1.5	Persistente Variablen	19
3.1.6	Symbole	20
3.2	Funktionsbaustein AnaMon	23
3.2.1	Kurzbeschreibung	23
3.2.2	Erweiterte Funktionen	23
3.2.3	FB-spezifischer Worst Quality Code	24
3.2.4	Persistente Variablen	25
3.2.5	Symbole	26
3.2.6	Faceplates	28
4	Analogwertoperation.....	30
4.1	Funktionsbaustein AnaMan	30
4.1.1	Kurzbeschreibung	30
4.1.2	Grundfunktionen	30
4.1.3	Funktionen	31
4.1.4	FB-spezifischer Worst Quality Code (WQC)	33
4.1.5	Persistente Variablen	33
4.1.6	Symbole	34
4.2	Funktionsbaustein AnaManInt	35
4.2.1	Kurzbeschreibung	35
4.2.2	Erweiterte Funktionen	35
4.2.3	FB-spezifischer Worst Quality Code (WQC)	35
4.2.4	Source Mode.....	36
4.2.5	Persistente Variablen	37
4.2.6	Symbole	38
4.2.7	Faceplates	39

5	Digitalwertanzeige	41
5.1	Funktionsbaustein DIntView	41
5.1.1	Kurzbeschreibung	41
5.1.2	Grundfunktionen	41
5.1.3	Funktionen	41
5.1.4	FB-spezifischer Worst Quality Code (WQC)	43
5.1.5	Persistente Variablen	43
5.1.6	Symbols	44
5.2	Funktionsbaustein DIntMon	47
5.2.1	Kurzbeschreibung	47
5.2.2	Erweiterte Funktionen	47
5.2.3	FB-spezifischer Worst Quality Code	48
5.2.4	Persistente Variablen	49
5.2.5	Symbole	50
5.2.6	Faceplates	52
6	Digitalwertoperation	54
6.1	Funktionsbaustein DIntMan	54
6.1.1	Kurzbeschreibung	54
6.1.2	Grundfunktionen	54
6.1.3	Funktionen	55
6.1.4	FB-spezifischer Worst Quality Code	57
6.1.5	Persistente Variablen	57
6.1.6	Symbole	58
6.2	Funktionsbaustein DIntManInt	59
6.2.1	Kurzbeschreibung	59
6.2.2	Erweiterte Funktionen	59
6.2.3	FB-spezifischer Worst Quality Code (WQC)	59
6.2.4	Source Mode	60
6.2.5	Persistente Variablen	61
6.2.6	Symbole	62
6.2.7	Faceplates	63
7	Binärwertanzeige	65
7.1	Funktionsbaustein BinView	65
7.1.1	Kurzbeschreibung	65
7.1.2	Grundfunktionen	65
7.1.3	Funktionen	65
7.1.4	FB-spezifischer Worst Quality Code (WQC)	66
7.1.5	Symbole	66
7.2	Funktionsbaustein BinMon	68
7.2.1	Kurzbeschreibung	68
7.2.2	Erweiterte Funktionen	68
7.2.3	FB-spezifischer Worst Quality Code	69
7.2.4	Persistente Variablen	69
7.2.5	Symbole	70
7.2.6	Faceplates	71

8	Binäroperationen.....	73
8.1	Funktionsbaustein BinMan	73
8.1.1	Kurzbeschreibung	73
8.1.2	Grundfunktionen	73
8.1.3	Funktionen	74
8.1.4	FB-spezifischer Worst Quality Code	75
8.1.5	Persistente Variablen	75
8.1.6	Symbole	76
8.2	Funktionsbaustein BinManInt	77
8.2.1	Kurzbeschreibung	77
8.2.2	Erweiterte Funktionen	77
8.2.3	FB-spezifischer Worst Quality Code (WQC)	77
8.2.4	Source Mode	77
8.2.5	Persistente Variablen	79
8.2.6	Symbole	80
8.2.7	Faceplates	81
9	Zeichenketteanzeige	82
9.1	Funktionsbaustein StrView	82
9.1.1	Kurzbeschreibung	82
9.1.2	Funktionen	82
9.1.3	FB-spezifischer Worst Quality Code	82
9.1.4	Persistente Variablen	82
9.1.5	Symbole	83
9.1.6	Faceplate.....	83
10	Regelungen	84
10.1	Funktionsbaustein PIDCtrl.....	84
10.1.1	Kurzbeschreibung	84
10.1.2	Grundfunktionen	84
10.1.3	Funktionen	85
10.1.4	Erweiterte Funktionen	87
10.1.5	FB-spezifischer Worst Quality Code (WQC)	88
10.1.6	Source Mode	89
10.1.7	Operation Mode	91
10.1.8	Persistente Variablen	93
10.1.9	Symbole	94
10.1.10	Faceplates	95
11	Bistabile Ventile	97
11.1	Funktionsbautein BinVlv	97
11.1.1	Kurzbeschreibung	97
11.1.2	Grundfunktionen	97
11.1.3	Funktionen	98
11.1.4	FB-spezifischer Worst Quality Code	100
11.1.5	Operation Mode	100
11.1.6	Persistente Variablen	102
11.1.7	Symbole	103
11.2	Funktionsbaustein MonBinVlv	105
11.2.1	Kurzbeschreibung	105

11.2.2	Erweiterte Funktionen	105
11.2.3	FB-spezifischer Worst Quality Code	107
11.2.4	Persistente Variablen	107
11.2.5	Symbole	108
11.2.6	Faceplates	110
12	Regelbare Ventile.....	112
12.1	Funktionsbaustein AnaVlv.....	112
12.1.1	Kurzbeschreibung	112
12.1.2	Grundfunktionen	112
12.1.3	Funktionen	113
12.1.4	FB-spezifischer Worst Quality Code	116
12.1.5	Source Mode.....	116
12.1.6	Operation Mode	118
12.1.7	Persistente Variablen	120
12.1.8	Symbole	121
12.2	Funktionsbaustein MonAnaVlv	124
12.2.1	Kurzbeschreibung	124
12.2.2	Erweiterte Funktionen	124
12.2.3	FB-spezifischer Worst Quality Code	127
12.2.4	Persistente Variablen	127
12.2.5	Symbole	129
12.2.6	Faceplates	131
13	Bistabile Motoren.....	133
13.1	Funktionsbaustein BinDrv	133
13.1.1	Kurzbeschreibung	133
13.1.2	Grundfunktionen	133
13.1.3	Funktionen	134
13.1.4	FB-spezifischer Worst Quality Code	137
13.1.5	Operation Mode	137
13.1.6	Persistente Variablen	139
13.1.7	Symbole	140
13.2	Funktionsbaustein MonBinDrv	142
13.2.1	Kurzbeschreibung	142
13.2.2	Erweiterte Funktionen	142
13.2.3	FB-spezifischer Worst Quality Code	144
13.2.4	Persistente Variablen	144
13.2.5	Symbole	146
13.2.6	Faceplates	148
14	Regelbare Motoren.....	150
14.1	Funktionsbaustein AnaDrv	150
14.1.1	Kurzbeschreibung	150
14.1.2	Grundfunktionen	150
14.1.3	Funktionen	151
14.1.4	FB-spezifischer Worst Quality Code	155
14.1.5	Source Mode.....	155
14.1.6	Operation Mode	157
14.1.7	Persistente Variablen	159

14.1.8	Symbole	160
14.2	Funktionsbaustein MonAnaDrv	162
14.2.1	Kurzbeschreibung	162
14.2.2	Erweiterte Funktionen	162
14.2.3	FB-spezifischer Worst Quality Code	165
14.2.4	Persistente Variablen	165
14.2.5	Symbole	167
14.2.6	Faceplates	169
15	Totalizer	171
15.1	Funktionsbaustein Totalizer	171
15.1.1	Kurzbeschreibung	171
15.1.2	Grundfunktionen	171
15.1.3	Funktionen	172
15.1.4	FB-spezifischer Worst Quality Code	174
15.1.5	Source Mode	174
15.1.6	Operation Mode	176
15.1.7	Persistente Variablen	177
15.1.8	Symbole	178
15.2	Funktionsbaustein MonTotalizer	180
15.2.1	Kurzbeschreibung	180
15.2.2	Funktionen	180
15.2.3	FB-spezifischer Worst Quality Code	181
15.2.4	Persistente Variablen	181
15.2.5	Symbole	183
15.2.6	Faceplates	184
16	MPA-Positioner	186
16.1	Funktionsbaustein DrvCtrl.....	186
16.1.1	Kurzbeschreibung	186
16.1.2	Grundfunktionen	186
16.1.3	Funktionen	187
16.1.4	FB-spezifischer Worst Quality Code	189
16.1.5	Source Mode	190
16.1.6	Operation Mode	191
16.1.7	Persistente Variablen	193
16.1.8	Symbole	195
16.1.9	Faceplates	197
17	Verriegelungsanzeige (Interlock)	199
17.1	Funktionsbaustein LockView4.....	199
17.1.1	Kurzbeschreibung	199
17.1.2	Grundfunktionen	199
17.1.3	Funktionen	199
17.1.4	FB-spezifischer Worst Quality Code	200
17.1.5	Persistente Variablen	200
17.1.6	Symbole	201
17.1.7	Faceplates	201
17.2	Funktionsbaustein LockView8.....	203
17.2.1	Kurzbeschreibung	203

17.2.2	Grundfunktionen	203
17.2.3	Funktionen	203
17.2.4	FB-spezifischer Worst Quality Code	204
17.2.5	Persistente Variablen	204
17.2.6	Symbole	205
17.2.7	Faceplates	205
17.3	Funktionsbaustein LockView16.....	207
17.3.1	Kurzbeschreibung	207
17.3.2	Grundfunktionen	207
17.3.3	Funktionen	207
17.3.4	FB-spezifischer Worst Quality Code	208
17.3.5	Persistente Variablen	208
17.3.6	Symbole	209
17.3.7	Faceplates	209
18	Services	212
18.1	SimpleSelfCompletingServiceControl	212
18.1.1	Kurzbeschreibung	212
18.1.2	Grundfunktionen	213
18.1.3	Funktionen	214
18.1.4	Erweiterte Funktionen	217
18.1.5	Source Mode	217
18.1.6	Operation Mode	218
18.2	SimpleContinuousServiceControl	221
18.2.1	Kurzbeschreibung	221
18.2.2	Grundfunktionen	222
18.2.3	Funktionen	222
18.2.4	Erweiterte Funktionen	226
18.2.5	Source Mode	226
18.2.6	Operation Mode	227
18.3	ServiceControl.....	230
18.3.1	Kurzbeschreibung	230
18.3.2	Grundfunktionen	231
18.3.3	Funktionen	232
18.3.4	Source Mode	235
18.3.5	Operation Mode	236
18.3.6	Symbol.....	238
18.3.7	Faceplates	239
19	Serviceprozeduren	241
19.1	ServiceProcedure	241
19.1.1	Kurzbeschreibung	241
19.1.2	Grundfunktionen	241
19.1.3	Funktionen	241
20	Analoge Dienstparameter	243
20.1	AnaConfigParam	243
20.1.1	Kurzbeschreibung	243
20.1.2	Grundfunktionen	243
20.1.3	Funktionen	244

20.1.4	Source Mode	245
20.1.5	Operation Mode	247
20.2	AnaProcedureParam	249
20.2.1	Kurzbeschreibung	249
20.2.2	Grundfunktionen	249
20.2.3	Funktionen	250
20.2.4	Source Mode	251
20.2.5	Operation Mode	253
20.3	AnaProcessValueIn	256
20.3.1	Kurzbeschreibung	256
20.3.2	Grundfunktionen	256
20.3.3	Funktionen	256
20.4	AnaProcessValueOut	258
20.4.1	Kurzbeschreibung	258
20.4.2	Grundfunktionen	258
20.4.3	Funktionen	259
20.4.4	FB-spezifischer Worst Quality Code (WQC)	261
20.4.5	Persistente Variablen	261
20.5	AnaReportValue	263
20.5.1	Kurzbeschreibung	263
20.5.2	Grundfunktionen	263
20.5.3	Funktionen	264
20.5.4	FB-spezifischer Worst Quality Code (WQC)	266
20.5.5	Persistente Variablen	266

21	Binäre Dienstparameter	268
21.1	BinConfigParam	268
21.1.1	Kurzbeschreibung	268
21.1.2	Grundfunktionen	268
21.1.3	Funktionen	269
21.1.4	Source Mode	270
21.1.5	Operation Mode	271
21.2	BinProcedureParam	273
21.2.1	Kurzbeschreibung	273
21.2.2	Grundfunktionen	273
21.2.3	Funktionen	274
21.2.4	Source Mode	275
21.2.5	Operation Mode	276
21.3	BinProcessValueIn	278
21.3.1	Kurzbeschreibung	278
21.3.2	Grundfunktionen	278
21.3.3	Funktionen	279
21.4	BinProcessValueOut	280
21.4.1	Kurzbeschreibung	280
21.4.2	Grundfunktionen	280
21.4.3	Funktionen	281
21.4.4	FB-spezifischer Worst Quality Code (WQC)	281
21.5	BinReportValue	282
21.5.1	Kurzbeschreibung	282
21.5.2	Grundfunktionen	282
21.5.3	Funktionen	283
21.5.4	FB-spezifischer Worst Quality Code (WQC)	283

22	Digitale Dienstparameter	284
22.1	DIntConfigParam	284
22.1.1	Kurzbeschreibung	284
22.1.2	Grundfunktionen	284
22.1.3	Funktionen	285
22.1.4	Source Mode	286
22.1.5	Operation Mode	288
22.2	DIntProcedureParam	290
22.2.1	Kurzbeschreibung	290
22.2.2	Grundfunktionen	290
22.2.3	Funktionen	291
22.2.4	Source Mode	292
22.2.5	Operation Mode	294
22.3	DIntProcessValueIn	296
22.3.1	Kurzbeschreibung	296
22.3.2	Grundfunktionen	296
22.3.3	Funktionen	297
22.4	DIntProcessValueOut	298
22.4.1	Kurzbeschreibung	298
22.4.2	Grundfunktionen	298
22.4.3	Funktionen	299
22.4.4	FB-spezifischer Worst Quality Code (WQC)	301
22.4.5	Persistente Variablen	301
22.5	DIntReportValue	303
22.5.1	Kurzbeschreibung	303
22.5.2	Grundfunktionen	303
22.5.3	Funktionen	304
22.5.4	FB-spezifischer Worst Quality Code (WQC)	306
22.5.5	Persistente Variablen	306
23	Zeichenketten-Dienstparameter	308
23.1	StringConfigParam	308
23.1.1	Kurzbeschreibung	308
23.1.2	Grundfunktionen	308
23.1.3	Funktionen	309
23.1.4	Source Mode	310
23.1.5	Operation Mode	311
23.2	StringProcedureParam	314
23.2.1	Kurzbeschreibung	314
23.2.2	Grundfunktionen	314
23.2.3	Funktionen	315
23.2.4	Source Mode	316
23.2.5	Operation Mode	317
23.3	StringProcessValueOut	320
23.3.1	Kurzbeschreibung	320
23.3.2	Grundfunktionen	320
23.3.3	Funktionen	321
23.4	StringReportValue	322
23.4.1	Kurzbeschreibung	322
23.4.2	Grundfunktionen	322
23.4.3	Funktionen	323

24	Parameterbausteine	324
24.1	Symbole	324
24.2	Faceplate Bool-Parameter	325
24.3	Faceplate Real-Parameter	326
24.3.2	Faceplate String-Parameter	327
25	Hilfsfunktionen & Enumeration.....	328
25.1	Hilfsfunktionen.....	328
25.1.1	Scaler	328
25.2	Enumeration.....	329
25.2.1	Enumeration WorstQualityCode	329
25.2.2	Enumeration UnitSettings	329
25.2.3	Enumeration ChannelAutOp	329
25.2.4	Enumeration LockLogic	329
25.2.5	Enumeration Command.....	330
25.2.6	Enumeration SMState	330

1 Allgemeine Hinweise

- Das PA Toolkit ist nach der aktuellen MTP-Spezifikation für die Funktionsbausteine und Services implementiert (Stand April 2020). Mit diesem Stand funktioniert auch der MTP-Exporter
- Alle Werte sind auf 2 Nachkommastellen genau berechnet. Abweichungen in weiteren Nachkommastellen sind softwarebedingt.
- Alle Werte die fest verbunden werden, können über ein Faceplate nicht mehr geändert werden
- Variablen zur Steuerung des Source Modes und Operation Mode dürfen ebenfalls nicht fest verbunden werden, da sonst das Handshake-Verfahren nicht mehr funktioniert.
- Für die Verschiebung der Faceplates werden Pointer verwendet. Nach einem Online-Change kann es passieren, dass Adressbereiche für Pointer überschrieben werden und die Verschiebung nicht mehr richtig funktioniert- aus diesem Grund sollte ein Online-Change vermieden werden.
- In der Visualisierung werden Gleitkommazahlen mit 5 Stellen vor dem Komma und 2 Nachkommastellen richtig dargestellt. Integer-Werte haben 5 Stellen zur Verfügung. Werden mehr Stellen verwendet, so kann es vorkommen, dass diese in der Visualisierung abgeschnitten werden.
- Es kann vorkommen, dass Einheiten mit sehr vielen Zeichen in den Faceplates aus Platzgründen abgeschnitten werden.

1.1 Versionsverwaltung

Es muss bei den Steuerungen unbedingt darauf geachtet werden, dass keine ältere Firmware verwendet wird. Durch das Lizenzierungsverfahren, welches auf Funktionen von Servicepack 12 zugreift, kann es sonst zu Ausnahmefehlern kommen.

1.1.1 Änderungshistorie

Release	Änderungen und Erweiterungen
1.0.1.1	Package für Codesys mit Funktionsblöcken, Symbolen, Faceplates, MTP-Exporter und Style für die Visualisierung
1.0.1.2	• Behebung von Fehlern im MTP-Exporter • Neue Ausgangsparameter an den Parameter Funktionsbausteine • Fehler in den Services behoben • Fehler in den Symbolen behoben
1.0.1.3	• Behebung von Fehlern im MTP-Exporter • Fehler in der Lizenzierung behoben • Fehler in den Services behoben • Fehler in der deutschen Dokumentation behoben • Englische Dokumentation verfügbar
2.0.5.0	• WIBU Lizenzierung • MTP Service Faceplate • Bugfixes

Tab. 1 Änderungshistorie

1.1.2 Release 1.0.1.1, 1.0.1.2, 1.0.1.3 und 2.0.5.0

Dieses Release wurde mit folgenden Versionen getestet und freigegeben:

Software/hardware	Version
CoDeSys	V3.5 SP12 Patch 6 pbF
Browser: Chrome	Version 81.0.4044.122 (Official Build) (64-Bit)
CPX-CEC-C1-V3	3.0.8.0.017190
CPX-CEC-S1-V3	3.0.8.0.017190
CPX-CEC-M1-V3	3.0.8.0.017190
CPX-E-CEC-M1-PN	1.1.22-73131e464.20190903
CPX-E-CEC-M1-EP	1.1.22-73131e464.20190903
CPX-E-CEC-C1-PN	1.1.22-73131e464.20190903
CPX-E-CEC-C1-EP	1.1.22-73131e464.20190903

Tab. 2 Releaseverwaltung

2 Lizenz für PA Toolkit

2.1 Kurzbeschreibung

Um das PA Toolkit dauerhaft betreiben zu können wird eine Lizenz benötigt.

Nach herunterladen des PA Toolkits kann die Applikation ohne jede zeitliche Begrenzung implementiert werden. Wird diese dann auf eine Steuerung heruntergeladen, wird überprüft, ob eine Lizenz vorhanden ist.

Jeder genutzte Funktionsblock benötigt eine Anzahl von Credits. Es gibt zwei Arten von Lizenzen. Die Credits Lizenz erlaubt Funktionsblöcke bis zur Anzahl der installierten Credits. Diese Lizenzen können mehrfach erworben und installiert werden, um die Gesamtanzahl der Credits zu erhöhen. Eine Flatrate Lizenz erlaubt die unbegrenzte Nutzung von Funktionsblöcken. Wenn die Anzahl der installierten Credits eine definierte Schwelle überschreitet, dann gilt das als Flatrate durch Credits.

2.2 Demolizenz

Ist keine Lizenz vorhanden startet die Demolizenz und die Steuerung kann für **2 Stunden** betrieben werden. Nach Ablauf der 2 Stunden muss die Steuerung neu gestartet werden bzw. die Applikation neu heruntergeladen werden. Ist die Demolizenz abgelaufen so wird jeder Funktionsbaustein in einen bestimmten Zustand gefahren. Auch über die Visualisierung können keine Einstellungen mehr vorgenommen werden. In den Symbolen wird ein entsprechender Hinweis angezeigt. In der folgenden Tabelle sind die Zustände jedes FBs beschrieben. Die Demolizenz entspricht einer Flatrate Lizenz.

Funktionsbaustein	Zustand
AnaView	V = VNormMin
AnaMon	V = VNormMin
AnaMan	VOut = SafeValue
AnaManInt	VOut = SafeValue
DIntView	V = VNormMin
DIntMon	V = VNormMin
DIntMan	VOut = SafeValue
DIntManInt	VOut = SafeValue
BinView	V = FALSE
BinMon	V = VAlmState
BinMan	VOut = SafeState
BinManInt	VOut = SafeState
StrView	TextOut = „“
PIDCtrl	Verhalten wie im Offline-Modus
BinVlv	Verhalten wie im Offline-Modus
MonBinVlv	Verhalten wie im Offline-Modus
AnaVlv	Verhalten wie im Offline-Modus
MonAnaVlv	Verhalten wie im Offline-Modus
BinDrv	Verhalten wie im Offline-Modus
MonBinDrv	Verhalten wie im Offline-Modus
AnaDrv	Verhalten wie im Offline-Modus
MonAnaDrv	Verhalten wie im Offline-Modus
LockView4/LockView8/LockView16	Out = FALSE
Totalizer	Stop Kommando wird ausgeführt
MonTotalizer	Stop Kommando wird ausgeführt
Service	Service wird in Hold gesetzt

DrvCtrl	GoTo100Vlv = 0 GoTo0Vlv = 0 CoarseGoTo100Vlv = 0 CoarseGoTo0Vlv = 0

Tab. 3 Zustände der einzelnen Funktionsbausteine nach Ablauf der Demoversion

Um das PA Toolkit dauerhaft auf einer Steuerung laufen zu lassen muss eine Lizenz über Festo SE & Co.KG erworben werden.

2.3 Lizenz erwerben und installieren

Lizenzen vom Typ Credits oder Flatrate können über die Festo Appworld (<https://www.festo.com/de/de/pa-toolkit.html>) erworben werden.

Installiert werden die Lizenzen mit der Festo Automation Suite. Sie ist über das Festo Support Portal (https://www.festo.com/net/de_de/SupportPortal/default.aspx?q=automation+suite&tab=4) verfügbar.

Wenn die Lizenz nach der Installation nicht direkt nutzbar ist, dann muss die Steuerung neu gestartet werden.

Um die installierte Lizenz, oder die Anzahl der benötigten und installierten Credits zu sehen muss wie folgt vorgegangen werden:

- Mindestens ein PA Toolkit Funktionsbaustein muss in einem Task laufen
- Visualisierung in CoDeSys erstellen
- Symbol hLicenseDefault in die Visualisierung ziehen
- Applikation herunterladen und starten
- Mit Klick auf das Lizenz-Symbol Faceplate öffnen
- Über den Button „Check License“ kann jederzeit überprüft werden, ob eine Lizenz für das PA Toolkit vorhanden ist.

2.4 Symbol

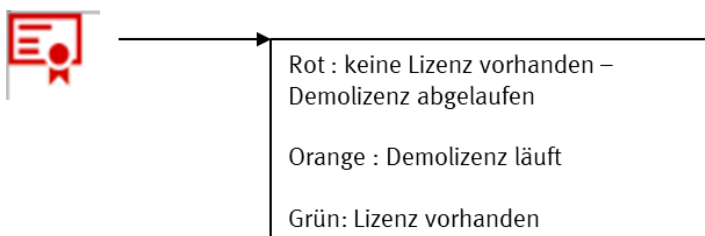


Fig. 1 Lizenz Symbol

2.5 Faceplate

2.5.1 Main

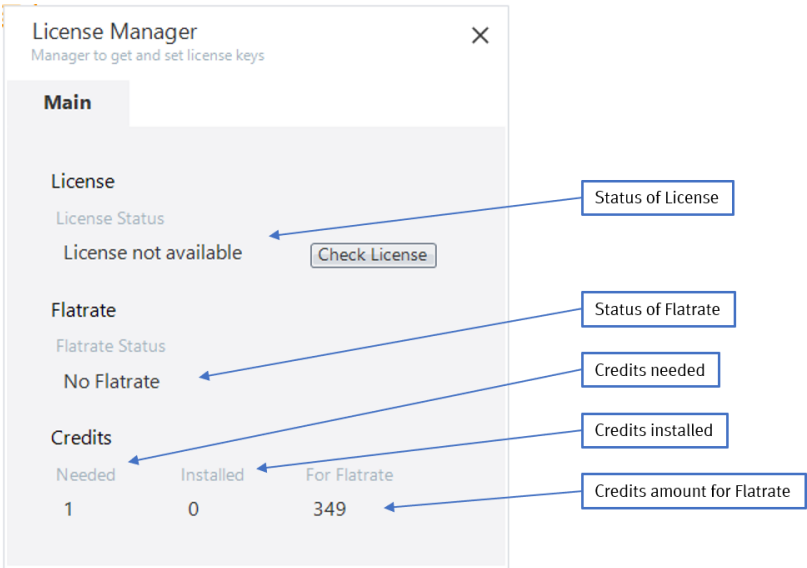


Fig. 2 Lizenz Main Faceplate

3 Analogwertanzeige

3.1 Funktionsbaustein AnaView

3.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaView bietet die Möglichkeit einen analogen Eingangswert (z.B. von einer analogen Eingangskarte) zu skalieren. Dieser umgewandelte Wert wird als Ausgang zur weiteren Verarbeitung zur Verfügung gestellt.

3.1.2 Grundfunktionen

3.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

3.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle möglichen Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 4 Worst Quality Code Enumeration

3.1.3 Funktionen

3.1.3.1 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer analogen Eingangskarte verbunden werden. Über die Normierungswerte „FVNormMin“ und „FVNormMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind ebenfalls Eingänge am FB.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert. Das bedeutet, dass der simulierte Wert nicht skaliert wird. Es wird nur überprüft, ob der simulierte Werte die Grenzwerte von „VNormMin“ und „VNormMax“ nicht unterschreitet bzw. überschreitet. Werden die Grenzwerte nicht eingehalten wird der entsprechende Grenzwert als „V“ ausgegeben.

3.1.3.2 Ausgangswert „V“

Der Eingangswert „FV“ wird mit Hilfe der Normierungswerte („FVNormMin“, „FVNormMax“, „VNormMin“ und „VNormMax“) in den Ausgangswert „V“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswert als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „V“ seine Normierungswerte („VNormMin“ und „VNormMax“) so wird der entsprechende Normierungswert als am Ausgang V“ ausgegeben.

3.1.3.3 Umrechnung Eingangswert „FV“ in Ausgangswert „V“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswert benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Eingang: FVNormMin und FVNormMax

Ausgang: VNormMin und VNormMax

3.1.3.4 Einheit für „V“

Für den Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

3.1.3.5 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von „FVNormMax“ und „FVNormMin“ nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

3.1.3.6 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable („VScIMin“ und „VScIMax“) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Ist der Eingang „FullScIEn“ auf TRUE gesetzt, so werden automatisch die Normierungswerte „VNormMin“ und „VNormMax“ als Anzeigegrenzen verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Normierungswerte gesetzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

3.1.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSImEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax - VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 5 AnaView spezifischer Worst Quality Code

3.1.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 6 AnaView persistente Variablen

3.1.6 Symbole

3.1.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max 16 Zeichen.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „VSim“ wird der Ausgangswert eingegeben.	Erscheint wenn VSimEn = TRUE ist.
Output value	Ausgangswert „V“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „VUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Measurement Lines (durchgezogene Linie) und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
Bargraph	Stellt die eingestellten Grenzwerte und den aktuellen Ausgangswert grafisch dar. Grenzwerte werden nur berücksichtigt, wenn Grenzwertüberwachung aktiviert wurde.	Grenzwerte müssen richtig eingestellt sein sonst wird der Bargraph fehlerhaft dargestellt. (z.B. Grenzwert High muss größer wie Grenzwert Low sein.) „VScMin“ und „VScMax“ skalieren den Bargraph.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 7 AnaView Symbolbeschreibung

3.1.6.2 hAnaViewDefault

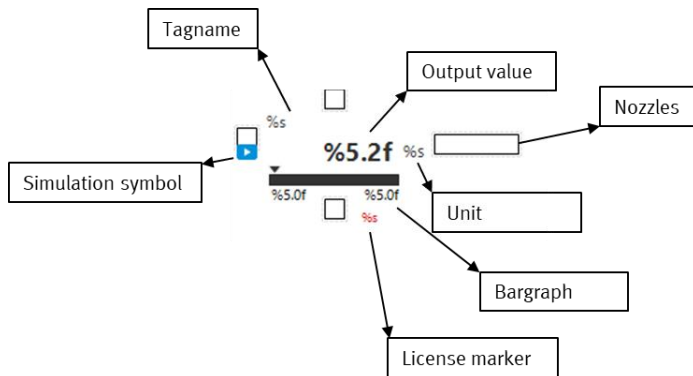


Fig. 3 Symbol hAnaViewDefault

3.1.6.3 hAnaViewValue

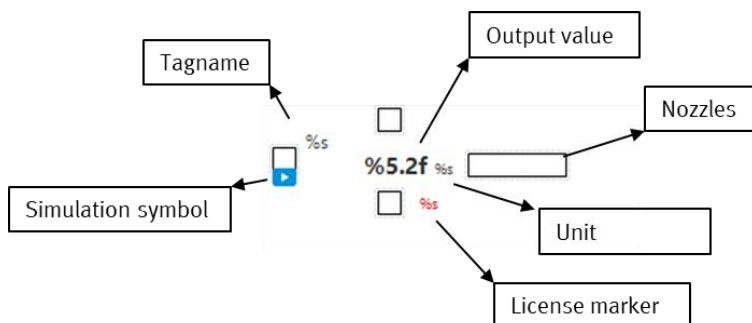


Fig. 4 Symbol hAnaViewValue

3.1.6.4 hAnaViewBargraph

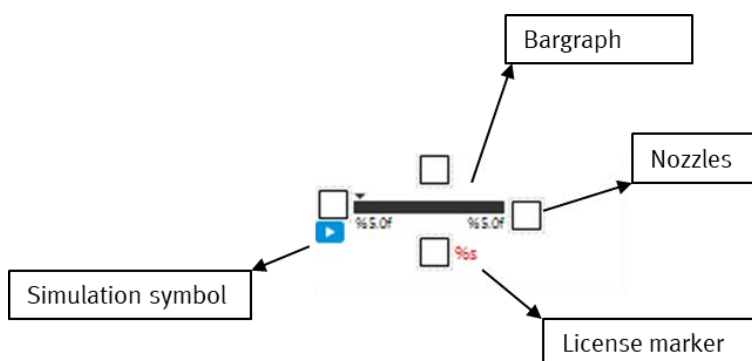


Fig. 5 Symbol hAnaViewBargraph

3.1.6.5 vAnaViewBargraph

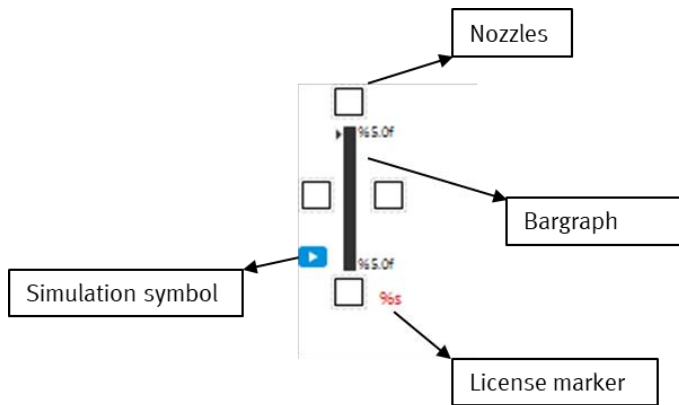


Fig. 6 Symbol vAnaViewBargraph

3.2 Funktionsbaustein AnaMon

3.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaMon erweitert den FB AnaView um die Grenzwertüberwachung.

3.2.2 Erweiterte Funktionen

3.2.2.1 Grenzwertüberwachung

Mit dieser Funktion wird der Ausgangswert „V“ überwacht. Es können verschiedene Grenzwerte eingegeben werden, die der Ausgangswert nicht unterschreiten bzw. überschreiten darf.

Dafür stehen 3 Überwachungskategorien zur Verfügung: Tolerance, Warning und Alarm.

Für jede Kategorie gibt es einen unteren Grenzwert (z.B. bei Alarm „VALLim“), der nicht unterschritten werden darf und einen oberen Grenzwert (z.B. bei Alarm „VAHLim“), der nicht überschritten werden darf. Die Grenzwerte stehen als Eingänge zur Verfügung und können voneinander unabhängig aktiviert werden (z. B. bei Alarm „VALEn“ und „VAHEn“). Zusätzlich kann über den Eingang „VALmTi“ eine Zeit vorgegeben werden. Wird eine Grenzwertverletzung erkannt, so muss diese erst die Zeit von „VALmTi“ anliegen, damit sie als Fehler erkannt wird. Andernfalls wird es als flatterndes Signal angesehen und es wird kein Alarm ausgelöst.

Ein Alarm wird über das zugehörige Ausgangsbit ausgegeben (z.B. bei Alarm „VALAct“ und „VAHAct“). Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

Werden Grenzwerte falsch gesetzt, (z.B. unterer Grenzwert > oberer Grenzwert) wird der WQC entsprechend gesetzt.

3.2.2.2 Zeit „VALmTi“

Die Variable „VALmTi“ wird in Sekunden angegeben und ist standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

3.2.2.3 Alarm quittieren

Liegt keine Grenzwertverletzung mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate (VALmAckOp) oder über den Eingang „VALmAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 8 AnaMon Alarmzustände

Hinweis:

- Der Eingang „VALmAckAut“ ist flankengesteuert und reagiert nur auf steigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarmer gleichzeitig quittiert.
- Es können auch mehrere Grenzwertverletzungen gleichzeitig anstehen und über die entsprechenden Ausgangsbits ausgegeben werden – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

3.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax – VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				Grenzwerte > VNormMax und/oder Grenzwerte < VNormMin (nur wenn sie aktiviert sind)
				Oberer Grenzwert < unterer Grenzwert (nur wenn sie aktiviert sind)
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Zustand bei jedem Zyklusstart

Tab. 9 AnaMon spezifischer Worst Quality Code

3.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)
VAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
VAHLim	REAL	0	Limit Value for Alarm High
VWHEn	BOOL	FALSE	Enables Warning High Limit (1: enabled 0: disabled)
VWHLim	REAL	0	Limit Value for Warning High
VTHEn	BOOL	FALSE	Enables Tolerance High Limit (1: enabled 0: disabled)
VTHLim	REAL	0	Limit Value for Tolerance High
VTLEn	BOOL	FALSE	Enables Tolerance Low Limit (1: enabled 0: disabled)
VTLLim	REAL	0	Limit Value for Tolerance Low
VWLEn	BOOL	FALSE	Enables Warning Low Limit (1: enabled 0: disabled)
VWLLim	REAL	0	Limit Value for Warning Low
VALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
VALLim	REAL	0	Limit Value for Alarm Low
VALmTi	REAL	0.5	Monitor Time to recognize limit error [s]

Tab. 10 AnaMon persistente Variablen

3.2.5 Symbole

3.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Alarm symbols	1 : Alarm ist aktiv (höchste Priorität) 2 : Warning alarm ist aktiv 3 : Tolerance alarm ist aktiv	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „VSim“ wird der Ausgangswert eingegeben.	Erscheint wenn VSimEn = TRUE ist.
Output value	Ausgangswert „V“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „VUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Measurement Lines (durchgezogene Linie) und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
Bargraph	Stellt die eingestellten Grenzwerte und den aktuellen Ausgangswert grafisch dar. Grenzwerte werden nur berücksichtigt, wenn Grenzwertüberwachung aktiviert wurde.	Grenzwerte müssen richtig eingestellt sein sonst wird der Bargraph fehlerhaft dargestellt. (z.B. Grenzwert High muss größer wie Grenzwert Low sein.) „VScIMin“ und „VScIMax“ skalieren den Bargraph.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 11 AnaMon Symbolbeschreibung

3.2.5.2 hAnaMonDefault

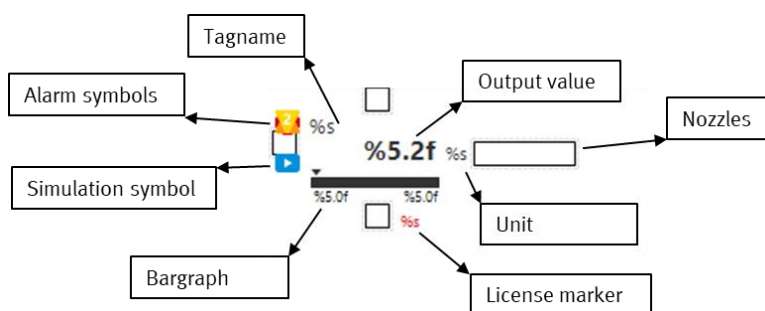


Fig. 7 Symbol hAnaMonDefault

3.2.5.3 hAnaMonValue

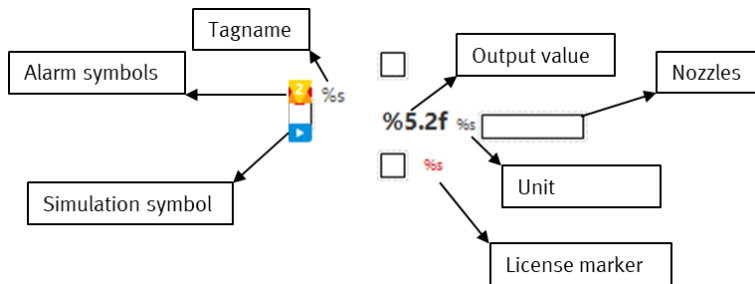


Fig. 8 Symbol hAnaMonValue

3.2.5.4 hAnaMonBargraph

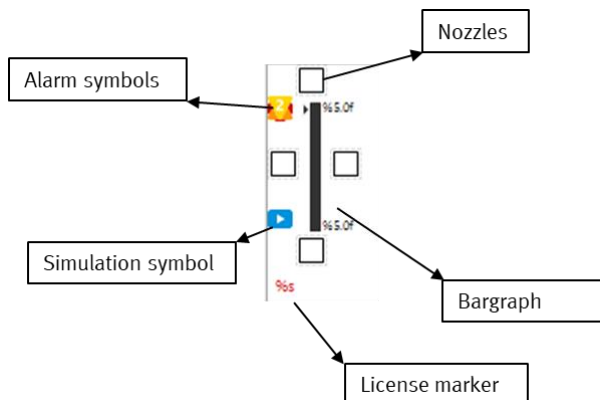


Fig. 9 Symbol hAnaMonBargraph

3.2.5.5 vAnaMonBargraph

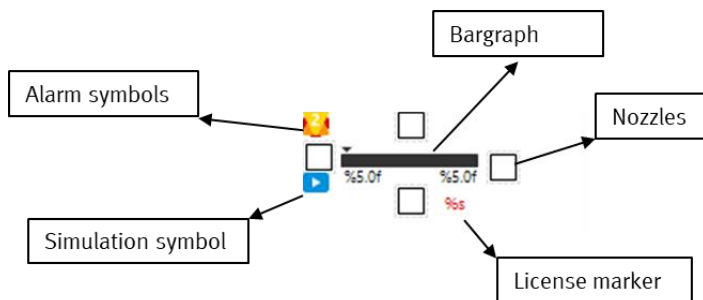


Fig. 10 Symbol vAnaMonBargraph

3.2.6 Faceplates

3.2.6.1 Main

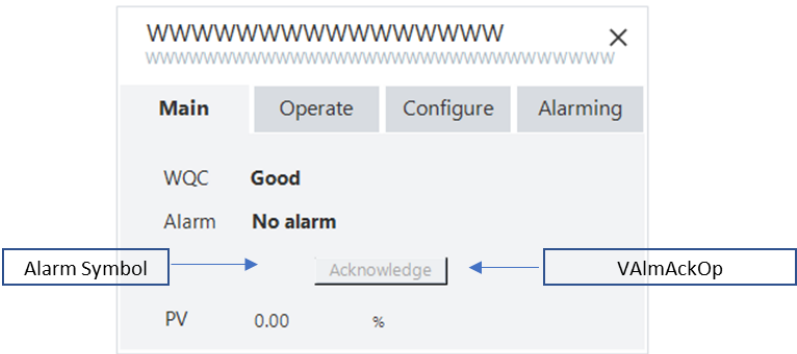


Fig. 11 AnaMon Main Faceplate

3.2.6.2 Operate

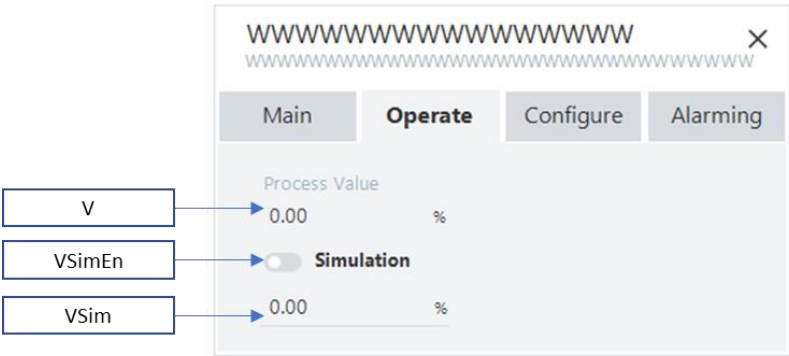


Fig. 12 AnaMon Operate Faceplate

3.2.6.3 Configure

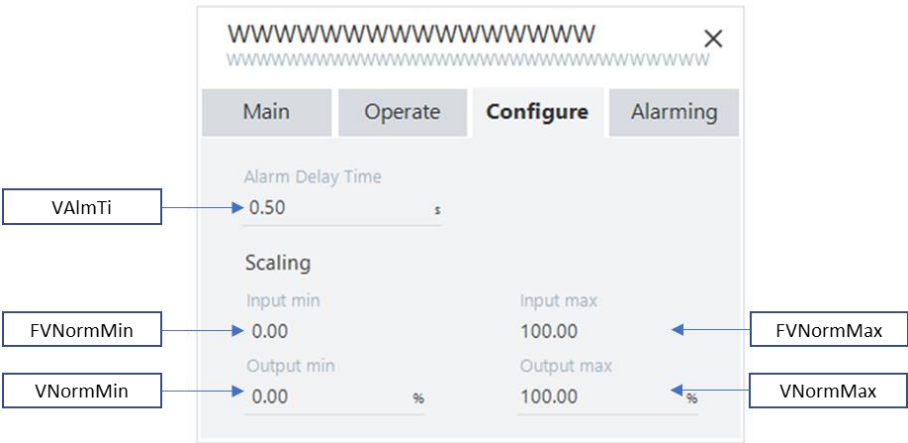


Fig. 13 AnaMon Configure Faceplate

3.2.6.4 Alarming

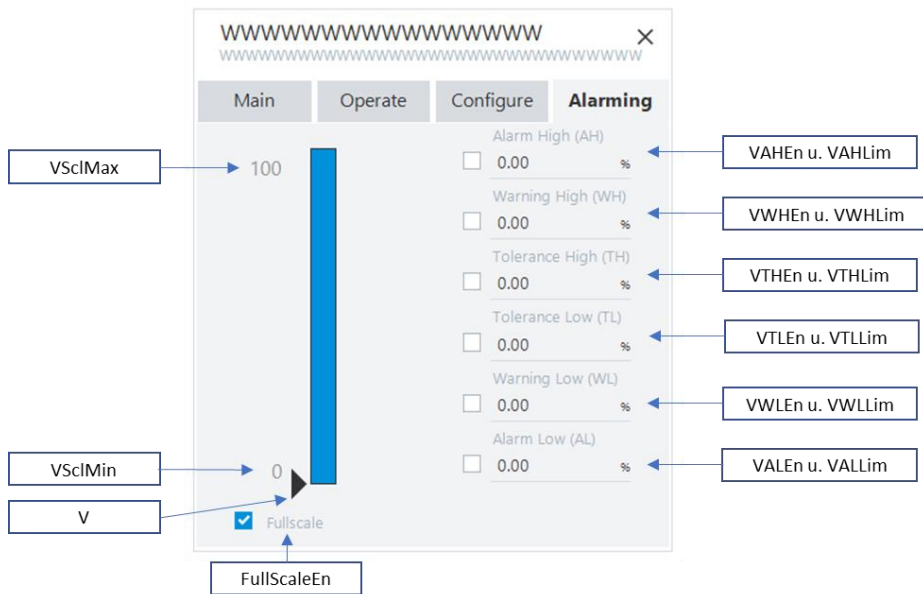


Fig. 14 AnaMon Alarming Faceplate

4 Analogwertoperation

4.1 Funktionsbaustein AnaMan

4.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaMan kann zum Übertragen eines analogen Wertes (Gleitkommazahl) aus der Prozessfeldebene genutzt werden. Das Signal kann in der Applikation weiterverarbeitet oder an eine analoge Ausgangskarte ausgegeben werden.

4.1.2 Grundfunktionen

4.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

4.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 12 Worst Quality Code Enumeration

4.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

4.1.3 Funktionen

4.1.3.1 Eingangswert „VMan“

Der Eingangswert wird durch die Variable „VMan“ repräsentiert und wird als OPC UA Variable zur Verfügung gestellt. Über die Normierungswerte „VMin“ und „VMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind als Eingänge am FB verfügbar.

4.1.3.2 Ausgangswert „VOut“

Der Eingangswert „VMan“ wird mit Hilfe der Normierungswerte („VMin“, „VMax“, „VOutMin“ und „VOutMax“) in den Ausgangswert „VOut“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswertes als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „VOut“ seine Normierungswerte („VOutMin“ und „VOutMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

4.1.3.3 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den Wert von „VMan“ aus jedoch ohne den Wert auf die Normierungswerte zu begrenzen.

4.1.3.4 Feedbacksignal „VFbk“

Der Eingang „VFbkSignal“ kann direkt mit dem Feedbacksignal aus dem Feld verbunden werden. Dieser wird zur weiteren Verarbeitung direkt an den Ausgang „VFbk“ weitergeleitet.

4.1.3.5 Umrechnung Eingangswert „VMan“ in Ausgangswert „VOut“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswert benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$V_{Out} = \frac{V_{OutMax} - V_{OutMin}}{V_{Max} - V_{Min}} * (V_{Man} - V_{Min}) + V_{OutMin}$$

Eingang: VMin und VMax

Ausgang: VOutMin und VOutMax

4.1.3.6 Einheit für „VMan“ und „VOut“

Für den Eingangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Ebenfalls kann eine Einheit auch für den Ausgangswert über den Eingang „VOutUnit“ eingestellt werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

4.1.3.7 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von „VOutMax“ und „VOutMin“ nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

4.1.3.8 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable („VScIMin“ und „VScIMax“) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

4.1.3.9 Sicherheitsposition

Über den Eingang „VSafeValue“ kann eine Sicherheitsposition angegeben werden. Im Falle eines Fehlers (siehe Lock-Funktionen) wird dieser Wert am Ausgang „VOut“ ausgegeben. Wird die Sicherheitsposition am „VOut“ ausgegeben, dann wird zusätzlich der Ausgang „SafePosAct“ auf TRUE gesetzt.

4.1.3.10 Lock-Funktionen

4.1.3.10.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

4.1.3.10.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird.

Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

4.1.3.10.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate („ResetOp“) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf steigende Flanken.

4.1.4 FB-spezifischer Worst Quality Code (WQC)

Nach der MTP-Spezifikation ist kein WQC in dieser Ausbaustufe vorgesehen.

4.1.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über **Applikation -> Add Object -> Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VMin	REAL	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	REAL	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	REAL	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 13 AnaMan persistente Variablen

4.1.6 Symbole

4.1.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Input value	„VMan“ wird ausgegeben.	Normierungswerte werden berücksichtigt.
Unit	Einheit „VUnit“ wird für den Eingangswert „VMan“ beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 14 AnaMan Symbolbeschreibung

4.1.6.2 hAnaManValue

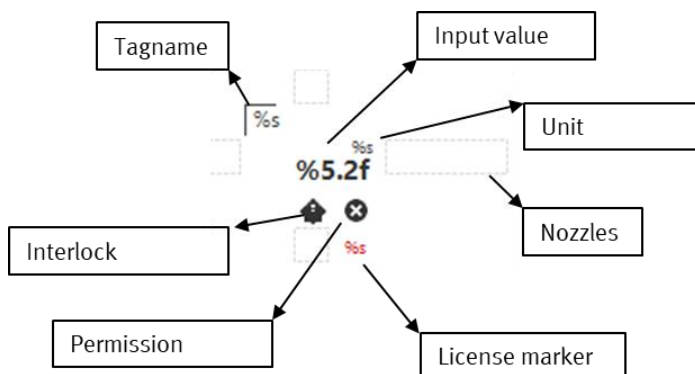


Fig. 15 Symbol hAnaManValue

4.2 Funktionsbaustein AnaManInt

4.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaManInt erweitert den FB AnaMan um einen weiteren Eingangswert (VInt), so dass von zwei unterschiedlichen Quellen ein Eingangswert gesetzt werden kann.

4.2.2 Erweiterte Funktionen

4.2.2.1 Eingangswert „VMan“ und „VInt“

Bei dem FB AnaMan stand der Eingang „VMan“ als OPC UA Variable zur Verfügung. Bei diesem FB steht ein weiterer Eingangswert „VInt“ als Eingang am FB zur Verfügung.

Über den Source Mode kann eingestellt werden, welcher Eingang für die Berechnung von „VOut“ herangezogen werden soll. Die Bedienung des Source Modes wird in einem extra Abschnitt erklärt.

Die eingestellten Normierungswerte („VMin“ und „VMax“) sind für beide Eingänge gültig, ebenfalls „VUnit“.

4.2.2.2 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt je nach Einstellung des Source Modes den Wert von „VMan“ oder „VInt“ aus, jedoch ohne den Wert auf die Normierungswerte zu begrenzen.

4.2.3 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	VMin > VMax
				VOutMin > VOutMax
				VMax-VMin = 0 ⇒ V = 0
				V < VOutMin ⇒ V = VOutMin
				V > VOutMax ⇒ V = VOutMax
				VScIMax < VScImin ⇒ VScIMax = VOutMax ⇒ VScImin = VOutMin
				VScIMax < VOutMin oder VScIMax > VOutMax ⇒ VScIMax = VOutMax
				VScImin < VOutMin oder VScImin > VOutMax ⇒ VScImin = VOutMin
				VScIMax - VScImin = 0 ⇒ VScIMax = VOutMax ⇒ VScImin = VOutMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 15 AnaManInt spezifischer Worst Quality Code

4.2.4 Source Mode

4.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang SrcIntAct auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit SrcManAct auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

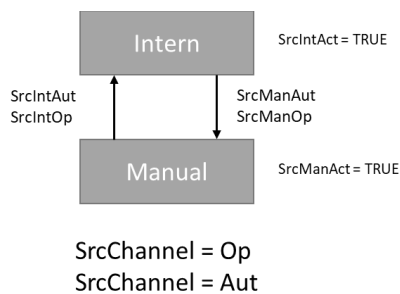


Fig. 16 Zustandsdiagramm Source Mode

4.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

4.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcManOp“, „SrcIntAut“ oder „SrcManAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt werden. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

4.2.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VMin	REAL	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	REAL	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	REAL	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 16 AnaManInt persistente Variablen

4.2.6 Symbole

4.2.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Input value	„VMan“ oder „VInt“ wird ausgegeben.	Abhängig vom Source Mode. Normierungswerte werden berücksichtigt.
Unit	Einheit „VUnit“ wird für den Eingangswert „VMan“ bzw. „VInt“ beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 17 AnaManInt Symbolbeschreibung

4.2.6.2 hAnaManIntValue

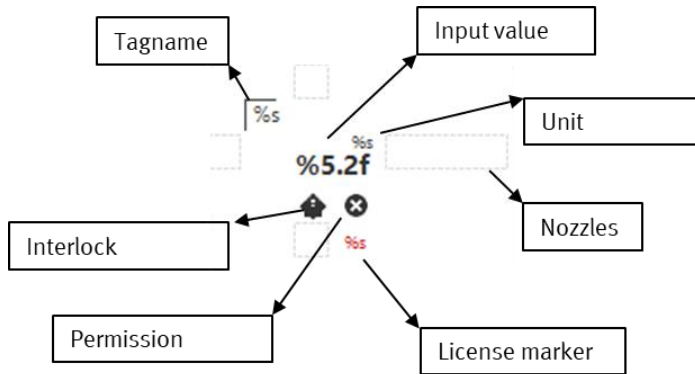


Fig. 17 Symbol hAnaManIntValue

4.2.7 Faceplates

4.2.7.1 Main

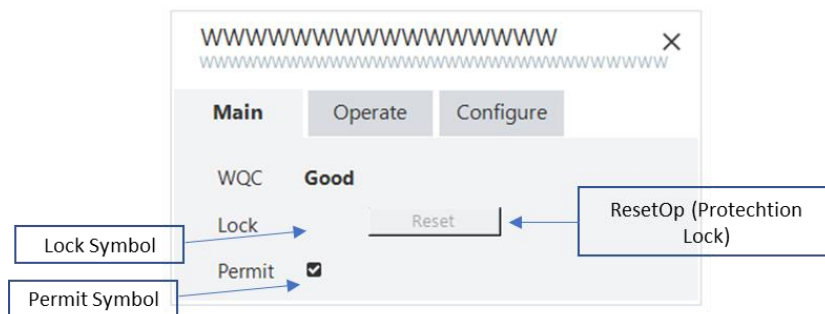


Fig. 18 AnaManInt Main Faceplate

4.2.7.2 Operate

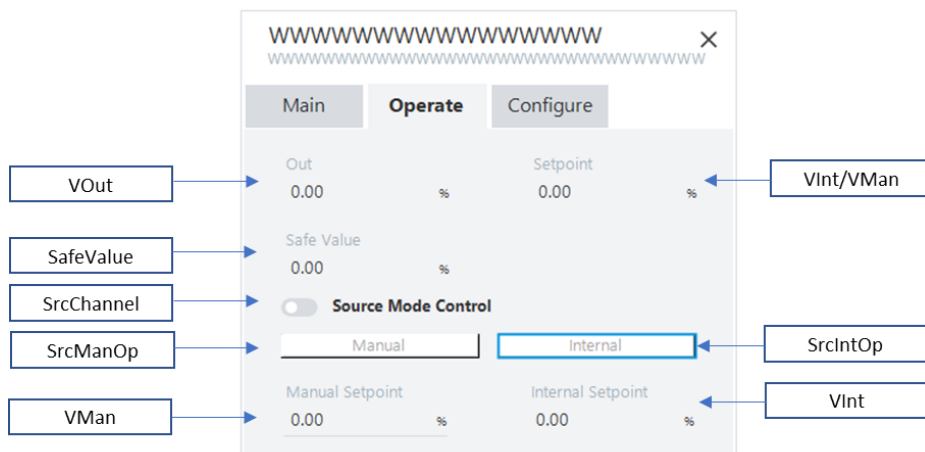


Fig. 19 AnaManInt Operate Faceplate

4.2.7.3 Configure

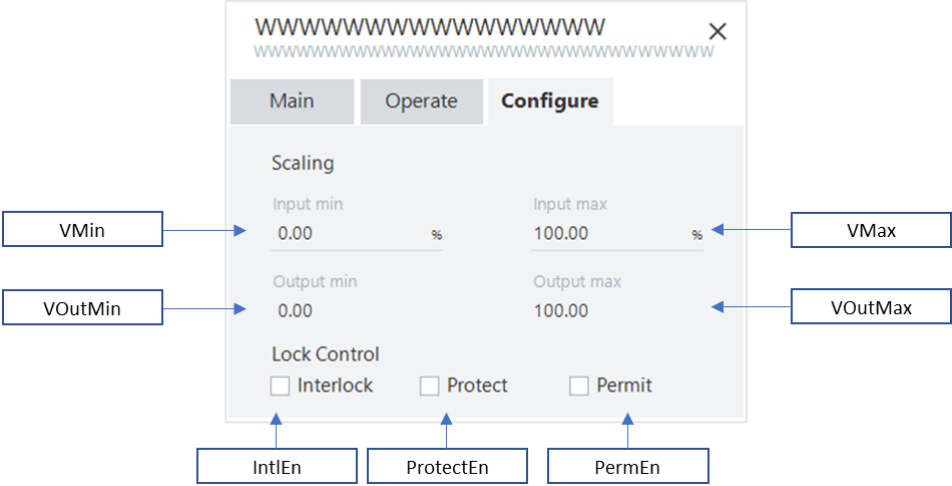


Fig. 20 AnaManInt Configure Faceplate

5 Digitalwertanzeige

5.1 Funktionsbaustein DIntView

5.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntView bietet die Möglichkeit einen analogen Eingangswert (z.B. von einer analogen Eingangskarte) zu skalieren. Dieser umgewandelte Wert wird als Ausgang zur weiteren Verarbeitung zur Verfügung gestellt. Alle Werte werden als Datentyp DINT verarbeitet.

5.1.2 Grundfunktionen

5.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

5.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 18 Worst Quality Code Enumeration

5.1.3 Funktionen

5.1.3.1 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer analogen Eingangskarte verbunden werden. Über die Normierungswerte „FVNormMin“ und „FVNormMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind ebenfalls Eingänge am FB.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert. Das bedeutet, dass der simulierte Wert nicht skaliert wird. Es wird nur überprüft, ob der simulierte Werte die Grenzwerte von „VNormMin“ und „VNormMax“ nicht unterschreitet bzw. überschreitet. Werden die Grenzwerte nicht eingehalten wird der entsprechende Grenzwert als „V“ ausgegeben.

5.1.3.2 Ausgangswert „V“

Der Eingangswert „FV“ wird mit Hilfe der Normierungswerte („FVNormMin“, „FVNormMax“, „VNormMin“ und „VNormMax“) in den Ausgangswert „V“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswert als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „V“ seine Normierungswerte („VNormMin“ und „VNormMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

Wird die Simulation verwendet wird als WQC „Simulated Value“ eingestellt. Der simulierte Wert wird nicht skaliert am Ausgang „V“ ausgegeben. D.h. es findet nur eine Grenzwertüberwachung und keine Umrechnung statt.

5.1.3.3 Umrechnung Eingangswert „FV“ in Ausgangswert „V“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswert benötigt, welche am FB als Eingänge eingegeben werden können.
Die Formel zur Umrechnung sieht wie folgt aus:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Eingang: FVNormMin und FVNormMax

Ausgang: VNormMin und VNormMax

5.1.3.4 Einheit für „V“

Für den Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

5.1.3.5 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von „FVNormMax“ und „FVNormMin“ nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

5.1.3.6 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable („VSclMin“ und „VSclMax“) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Ist der Eingang „FullSclEn“ auf TRUE gesetzt so werden automatisch die Normierungswerte „VNormMin“ und „VNormMax“ als Anzeigegrenzen verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Normierungswerte gesetzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

5.1.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax - VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 19 DIntView spezifischer Worst Quality Code

5.1.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden sollen.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 20 DIntView persistente Variablen

5.1.6 Symbols

5.1.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „VSim“ wird der Ausgangswert eingegeben.	Erscheint wenn VSimEn = TRUE ist.
Output value	Ausgangswert „V“ wird dargestellt.	Max. 5 Stellen werden angezeigt.
Unit	Einheit „VUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Measurement Lines (durchgezogene Linie) und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
Bargraph	Stellt die eingestellten Grenzwerte und den aktuellen Ausgangswert grafisch dar. Grenzwerte werden nur berücksichtigt, wenn Grenzwertüberwachung aktiviert wurde.	Grenzwerte müssen richtig eingestellt sein sonst wird der Bargraph fehlerhaft dargestellt. (z.B. Grenzwert High muss größer wie Grenzwert Low sein.) „VScMin“ und „VScMax“ skalieren den Bargraph.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 21 DIntView Symbolbeschreibung

5.1.6.2 hDIntViewDefault

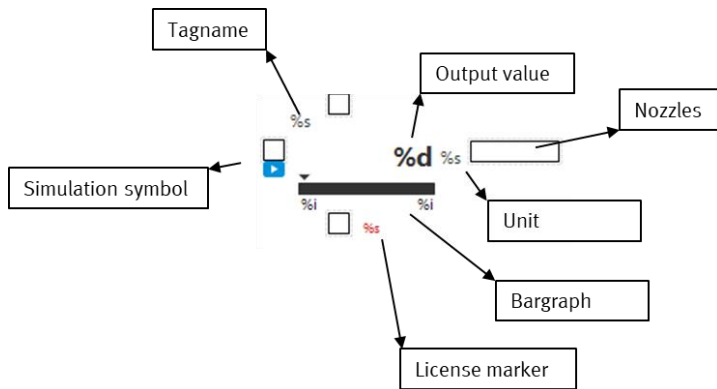


Fig. 21 Symbol hDIntViewDefault

5.1.6.3 hDIntViewValue

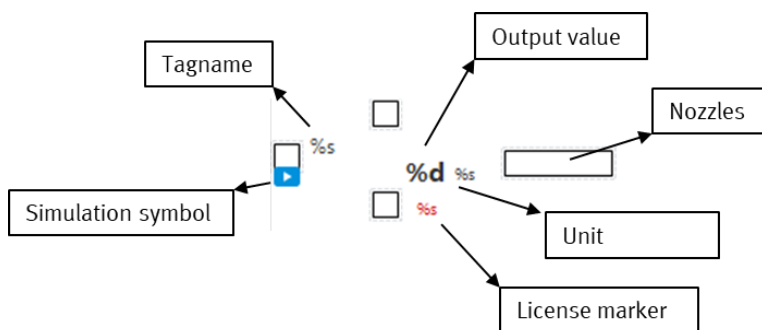


Fig. 22 Symbol hDIntViewValue

5.1.6.4 hDIntViewBargraph

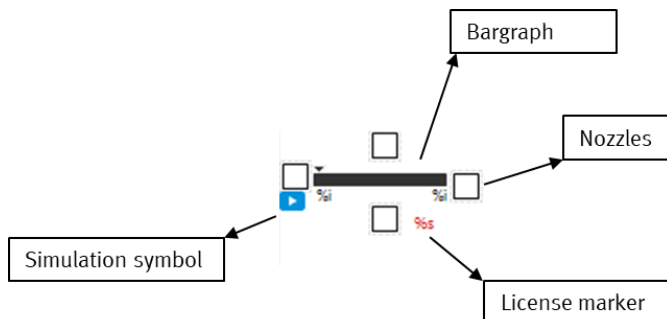


Fig. 23 Symbol hDIntViewBargraph

5.1.6.5 vDIntViewBargraph

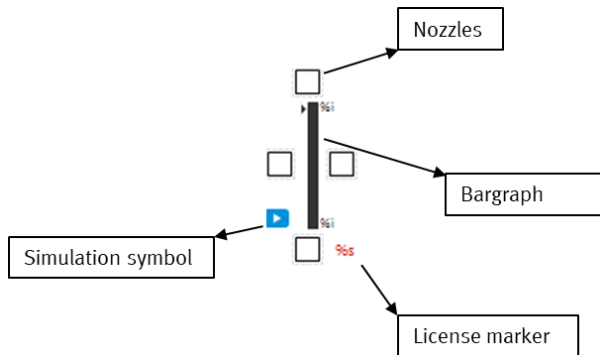


Fig. 24 Symbol vDIntViewBargraph

5.2 Funktionsbaustein DIntMon

5.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntMon erweitert den FB DIntView um die Grenzwertüberwachung.

5.2.2 Erweiterte Funktionen

5.2.2.1 Grenzwertüberwachung

Mit dieser Funktion wird der Ausgangswert „V“ überwacht. Es können verschiedene Grenzwerte eingegeben werden, die der Ausgangswert nicht unterschreiten bzw. überschreiten darf.

Dafür stehen 3 Überwachungskategorien zur Verfügung: Tolerance, Warning und Alarm.

Für jede Kategorie gibt es einen unteren Grenzwert (z.B. bei Alarm „VALLim“), der nicht unterschritten werden darf und einen oberen Grenzwert (z.B. bei Alarm „VAHLim“), der nicht überschritten werden darf. Die Grenzwerte stehen als Eingänge zur Verfügung und können voneinander unabhängig aktiviert werden (z. B. bei Alarm „VALEn“ und „VAHEn“). Zusätzlich kann über den Eingang „VALmTi“ eine Zeit vorgegeben werden. Wird eine Grenzwertverletzung erkannt, so muss diese erst die Zeit von „VALmTi“ anliegen, damit sie als Fehler erkannt wird. Andernfalls wird es als flatterndes Signal angesehen und es wird kein Alarm ausgelöst.

Ein Alarm wird über das zugehörige Ausgangsbit ausgegeben (z.B. bei Alarm „VALAct“ und „VAHAct“). Wird eine Visualisierung in CoDeSys verwendet so wird der entsprechende Alarm im Symbol angezeigt.

Werden Grenzwerte falsch gesetzt, (z.B. unterer Grenzwert > oberer Grenzwert) wird der WQC entsprechend gesetzt.

5.2.2.2 Zeit „VALmTi“

Die Variable „VALmTi“ wird in Sekunden angegeben und ist standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

5.2.2.3 Alarm quittieren

Liegt keine Grenzwertverletzung mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („VALmAckOp“) oder über den Eingang „VALmAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 22 Alarmzustände im Symbol

Hinweis:

- Der Eingang „VALmAckAut“ ist flankengesteuert und reagiert nur auf steigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarmer gleichzeitig quittiert.
- Es können auch mehrere Grenzwertverletzungen gleichzeitig anstehen und über die entsprechenden Ausgangsbits ausgegeben werden – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

5.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax – VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				Grenzwerte > VNormMax und/oder Grenzwerte < VNormMin (nur wenn sie aktiviert sind)
				Oberer Grenzwert < unterer Grenzwert (nur wenn sie aktiviert sind)
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 23 DIntMon spezifischer Worst Quality Code

5.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)
VAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
VAHLim	DINT	0	Limit Value for Alarm High
VWHEn	BOOL	FALSE	Enables Warning High Limit (1: enabled 0: disabled)
VWHLim	DINT	0	Limit Value for Warning High
VTHEn	BOOL	FALSE	Enables Tolerance High Limit (1: enabled 0: disabled)
VTHLim	DINT	0	Limit Value for Tolerance High
VTLEn	BOOL	FALSE	Enables Tolerance Low Limit (1: enabled 0: disabled)
VTLLim	DINT	0	Limit Value for Tolerance Low
VWLEn	BOOL	FALSE	Enables Warning Low Limit (1: enabled 0: disabled)
VWLLim	DINT	0	Limit Value for Warning Low
VALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
VALLim	DINT	0	Limit Value for Alarm Low
VALmTi	REAL	0.5	Monitor Time to recognize a limit error [s]

Tab. 24 DIntMon persistente Variablen

5.2.5 Symbole

5.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Alarm symbols	1 : Alarm ist aktiv (höchste Priorität) 2 : Warning alarm ist aktiv 3 : Tolerance alarm ist aktiv	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „VSim“ wird der Ausgangswert eingegeben.	Erscheint wenn VSimEn = TRUE ist.
Output value	Ausgangswert „V“ wird dargestellt.	Max. 5 Stellen werden angezeigt.
Unit	Einheit „VUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Measurement Lines (durchgezogene Linie) und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
Bargraph	Stellt die eingestellten Grenzwerte und den aktuellen Ausgangswert grafisch dar. Grenzwerte werden nur berücksichtigt, wenn Grenzwertüberwachung aktiviert wurde.	Grenzwerte müssen richtig eingestellt sein sonst wird der Bargraph fehlerhaft dargestellt. (z.B. Grenzwert High muss größer wie Grenzwert Low sein.) „VScIMin“ und „VScIMax“ skalieren den Bargraph
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 25 DIntMon Symbolbeschreibung

5.2.5.2 hDIntMonDefault

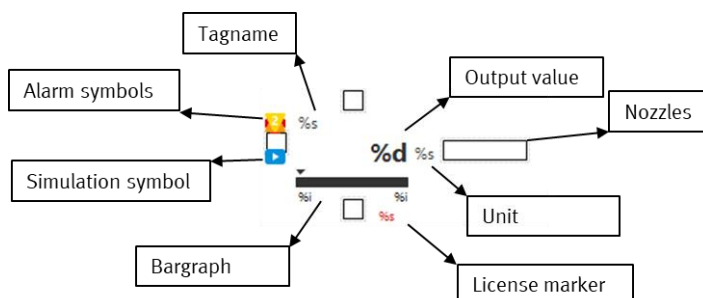


Fig. 25 Symbol hDIntMonDefault

5.2.5.3 hDIntMonValue

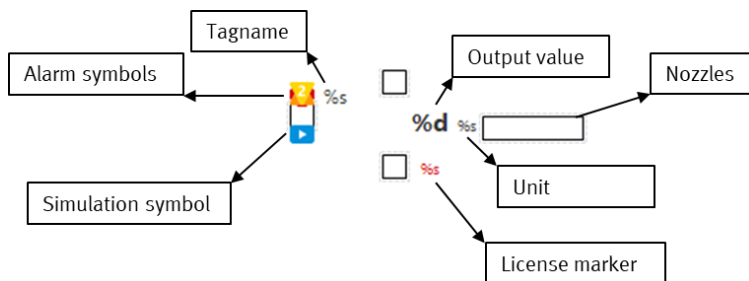


Fig. 26 Symbol hDIntMonValue

5.2.5.4 hDIntMonBargraph

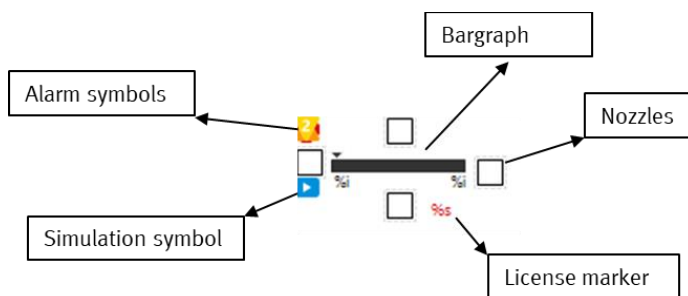


Fig. 27 Symbol hDIntMonBargraph

5.2.5.5 vDIntMonBargraph

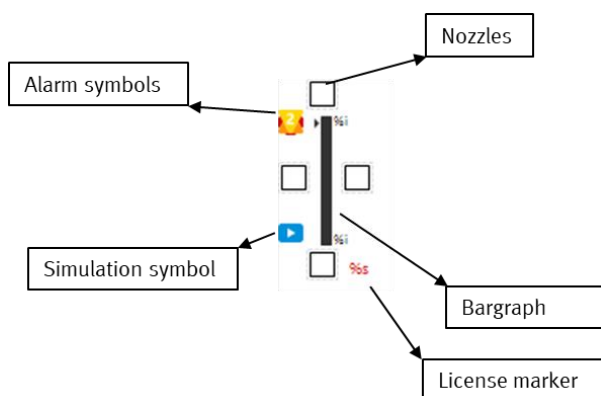


Fig. 28 Symbol vDIntMonBargraph

5.2.6 Faceplates

5.2.6.1 Main

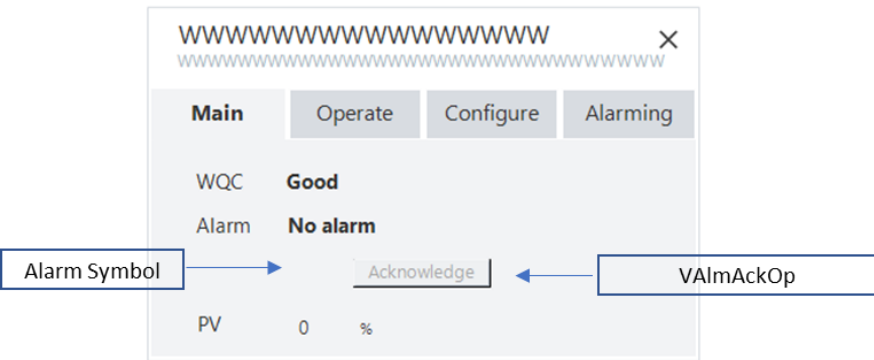


Fig. 29 DIntMon Main Faceplate

5.2.6.2 Operate

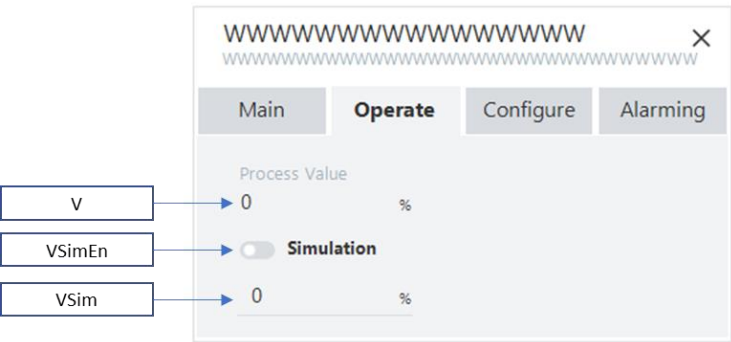


Fig. 30 DIntMon Operate Faceplate

5.2.6.3 Configure

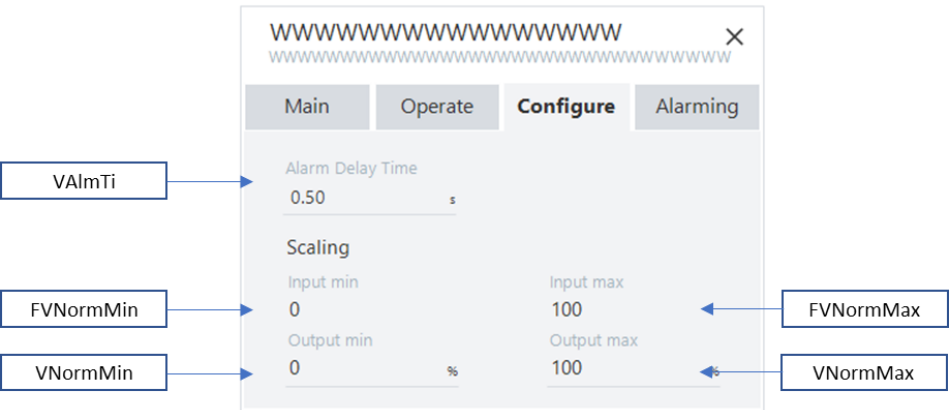


Fig. 31 DIntMon Configure Faceplate

5.2.6.4 Alarming

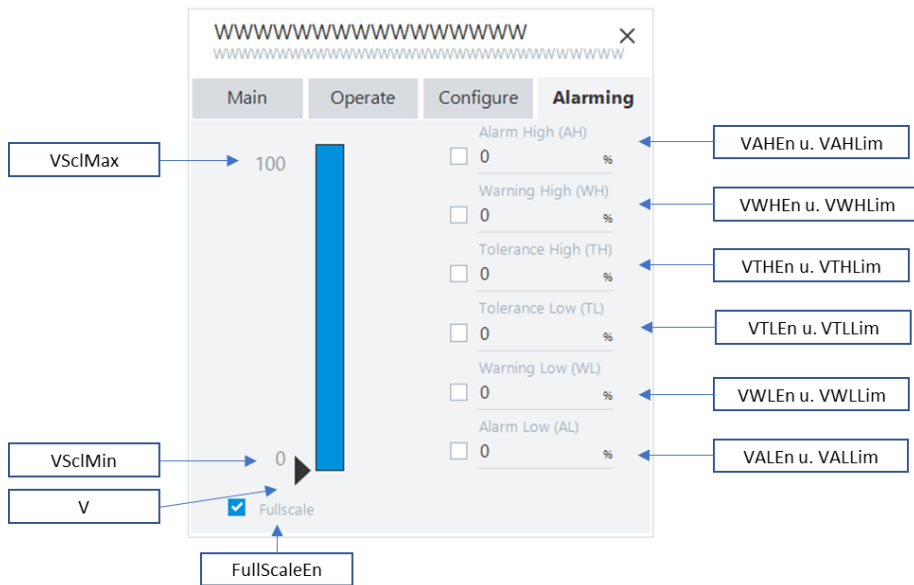


Fig. 32 DIntMon Alarming Faceplate

6 Digitalwertoperation

6.1 Funktionsbaustein DIntMan

6.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntMan kann zum Übertragen eines analogen Wertes (Double-Integer) aus der Prozessfeldebene genutzt werden. Das Signal kann in der Applikation weiterverarbeitet oder an eine analoge Ausgangskarte ausgegeben werden.

6.1.2 Grundfunktionen

6.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

6.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 26 Worst Quality Code Enumeration

6.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

6.1.3 Funktionen

6.1.3.1 Eingangswert „VMan“

Der Eingangswert wird durch die Variable „VMan“ repräsentiert und wird als OPC UA Variable zur Verfügung gestellt. Über die Normierungswerte „VMin“ und „VMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind als Eingänge am FB verfügbar.

6.1.3.2 Ausgangswert „VOut“

Der Eingangswert „VMan“ wird mit Hilfe der Normierungswerte („VMin“, „VMax“, „VOutMin“ und „VOutMax“) in den Ausgangswert „VOut“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswertes als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „VOut“ seine Normierungswerte („VOutMin“ und „VOutMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

6.1.3.3 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den Wert von „VMan“ aus jedoch ohne den Wert auf die Normierungswerte zu begrenzen.

6.1.3.4 Feedbacksignal „VFbk“

Der Eingang „VFbkSignal“ kann direkt mit dem Feedbacksignal aus dem Feld verbunden werden. Dieser wird zur weiteren Verarbeitung direkt an den Ausgang „VFbk“ weitergeleitet.

6.1.3.5 Umrechnung Eingangswert „VMan“ in Ausgangswert „VOut“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswerte benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$VOut = \frac{VOutMax - VOutMin}{VMax - VMin} * (VMan - VMin) + VOutMin$$

Eingang: VMin und VMax

Ausgang: VOutMin und VOutMax

6.1.3.6 Einheit für „VMan“ und „VOut“

Für den Eingangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Ebenfalls kann eine Einheit auch für den Ausgangswert über den Eingang „VOutUnit“ eingestellt werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

6.1.3.7 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von „VOutMax“ und „VOutMin“ nicht 0 ergeben, da nicht durch 0 geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

6.1.3.8 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable („VScIMin“ und „VScIMax“) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

6.1.3.9 Sicherheitsposition

Über den Eingang „VSafeValue“ kann eine Sicherheitsposition angegeben werden. Im Falle eines Fehlers (siehe Lock-Funktionen) wird dieser Wert am Ausgang „VOut“ ausgegeben. Wird die Sicherheitsposition am „VOut“ ausgegeben, dann wird zusätzlich der Ausgang „SafePosAct“ auf TRUE gesetzt.

6.1.3.10 Lock-Funktionen

6.1.3.10.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

6.1.3.10.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

6.1.3.10.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate (ResetOp) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf steigende Flanken.

6.1.4 FB-spezifischer Worst Quality Code

Nach der MTP-Spezifikation ist kein WQC in dieser Ausbaustufe vorgesehen.

6.1.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VMin	DINT	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	DINT	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	DINT	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 27 DintMan persistente Variablen

6.1.6 Symbole

6.1.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Input value	„VMan“ wird ausgegeben.	Normierungswerte werden berücksichtigt.
Unit	Einheit „VUnit“ wird für den Eingangswert „VMan“ beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 28 DIntMan Symbolbeschreibung

6.1.6.2 hDIntManValue

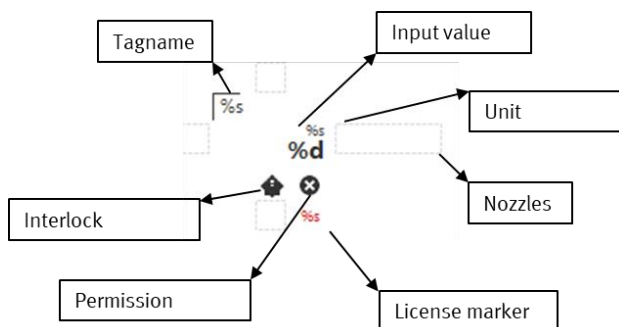


Fig. 33 Symbol hDIntManValue

6.2 Funktionsbaustein DIntManInt

6.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntManInt erweitert den FB DIntMan um einen weiteren Eingangswert (VInt), so dass von zwei unterschiedlichen Quellen ein Eingangswert gesetzt werden kann.

6.2.2 Erweiterte Funktionen

6.2.2.1 Eingangswert „VMan“ und „VInt“

Bei dem FB DIntMan stand der Eingang „VMan“ als OPC UA Variable zur Verfügung. Bei diesem FB steht ein weiterer Eingangswert „VInt“ als Eingang am FB zur Verfügung.

Über den Source Mode kann eingestellt werden, welcher Eingang für die Berechnung von „VOut“ herangezogen werden soll. Die Bedienung des Source Modes wird in einem extra Abschnitt erklärt.

Die eingestellten Normierungswerte (VMin und VMax) sind für beide Eingänge gültig, ebenfalls VUnit.

6.2.2.2 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt je nach Einstellung des Source Modes den Wert von „VMan“ oder „VInt“ aus jedoch ohne den Wert auf die Normierungswerte zu begrenzen.

6.2.3 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	VMin > VMax
				VOutMin > VOutMax
				VMax-VMin = 0 ⇒ V = 0
				V < VOutMin ⇒ V = VOutMin
				V > VOutMax ⇒ V = VOutMax
				VScIMax < VScImin ⇒ VScIMax = VOutMax ⇒ VScImin = VOutMin
				VScIMax < VOutMin oder VScIMax > VOutMax ⇒ VScIMax = VOutMax
				VScImin < VOutMin oder VScImin > VOutMax ⇒ VScImin = VOutMin
				VScIMax - VScImin = 0 ⇒ VScIMax = VOutMax ⇒ VScImin = VOutMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 29 DIntManInt spezifischer Worst Quality Code

6.2.4 Source Mode

6.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang SrcIntAct auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit SrcManAct auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

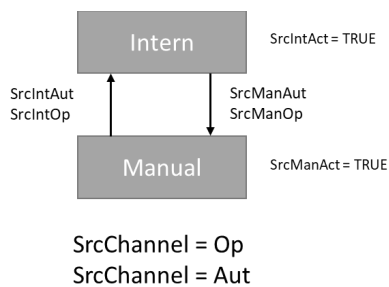


Fig. 34 Zustandsdiagramm Source Mode

6.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

6.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcManOp“, „SrcIntAut“ oder „SrcManAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt werden. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

6.2.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VMin	DINT	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	DINT	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	DINT	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 30 DIntManInt persistente Variablen

6.2.6 Symbole

6.2.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Input value	„VMan“ oder „VInt“ wird ausgegeben.	Abhängig vom Source Mode. Normierungswerte werden berücksichtigt.
Unit	Einheit „VUnit“ wird für den Eingangswert „VMan“ bzw. „VInt“ beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 31 DIntManInt Symbolbeschreibung

6.2.6.2 hDIntManIntValue

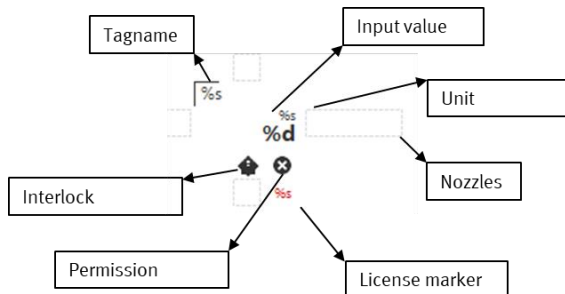


Fig. 35 Symbol hDIntManIntValue

6.2.7 Faceplates

6.2.7.1 Main

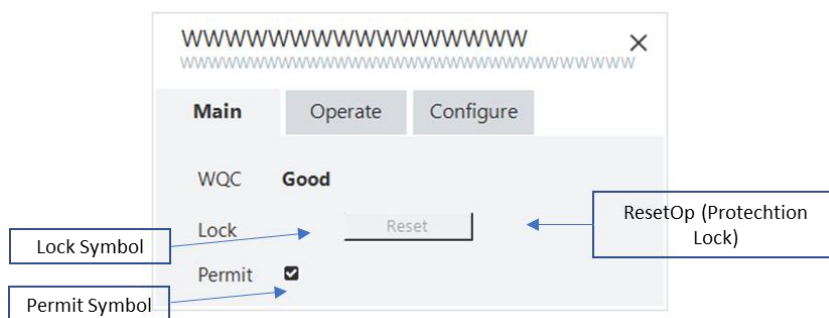


Fig. 36 DIntManInt Main Faceplate

6.2.7.2 Operate

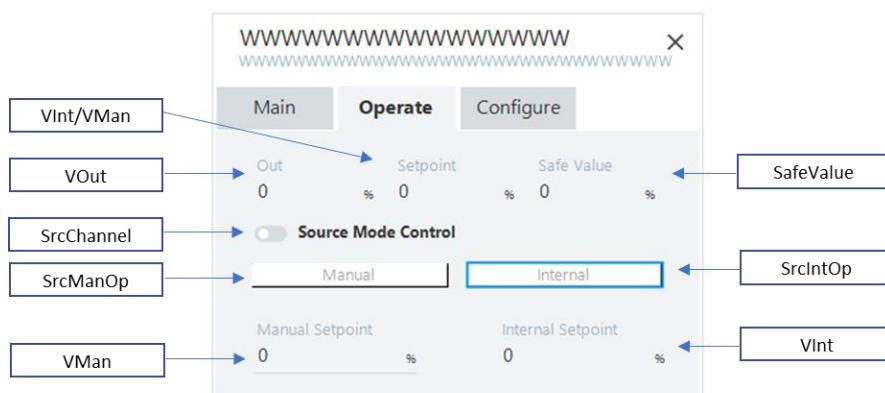


Fig. 37 DIntManInt Operate Faceplate

6.2.7.3 Configure

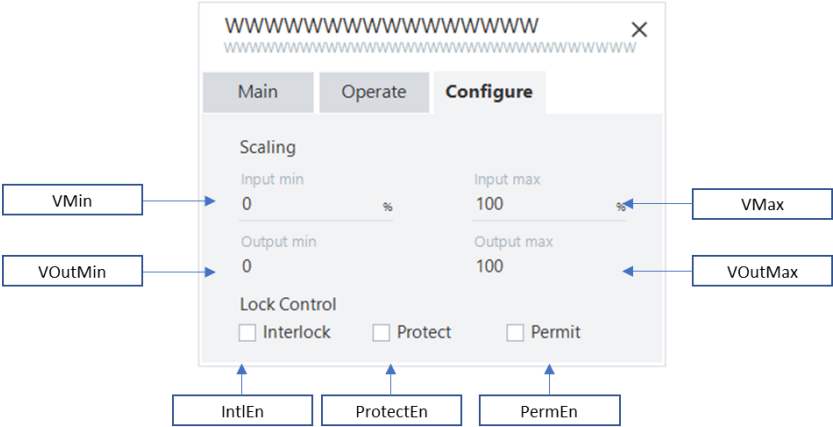


Fig. 38 DIntManInt Configure Faceplate

7 Binärwertanzeige

7.1 Funktionsbaustein BinView

7.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinView bietet die Möglichkeit einen binären (digitalen) Eingangswert (z.B. von einer binären Eingangskarte) für die weitere Verarbeitung an einen Ausgang auszugeben.

7.1.2 Grundfunktionen

7.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

7.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 32 Worst Quality Code Enumeration

7.1.3 Funktionen

7.1.3.1 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer digitalen Eingangskarte verbunden werden.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert.

7.1.3.2 Statustext „VState0“ und „VState1“

Am FB stehen die Eingänge „State0“ und „State1“ als zwei String-Variable zur Verfügung. An diesen Eingängen können Texte für die beiden Zustände hinterlegt werden. Diese beschreiben zusätzlich die Zustände in Worten. So kann z.B. für den Wert TRUE der Eingang VState1 mit dem Text „Tank ist voll“ belegt werden.

Für jeden Statustext stehen 24 Zeichen zur Verfügung.

7.1.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSIMEn (= TRUE)
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 33 BinView spezifischer Worst Quality Code

7.1.5 Symbole

7.1.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „VSIM“ wird der Ausgangswert eingegeben.	Erscheint wenn VSIMEn = TRUE ist.
Output value	Ausgangswert „V“ als Text.	Je nach Wert wird Text von „State0“ oder „State1“ angezeigt.
Nozzles	Nozzles zur Verbindung von Measurement Lines (durchgezogene Linie) und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 34 BinView Symbolbeschreibung

7.1.5.2 hBinViewValue

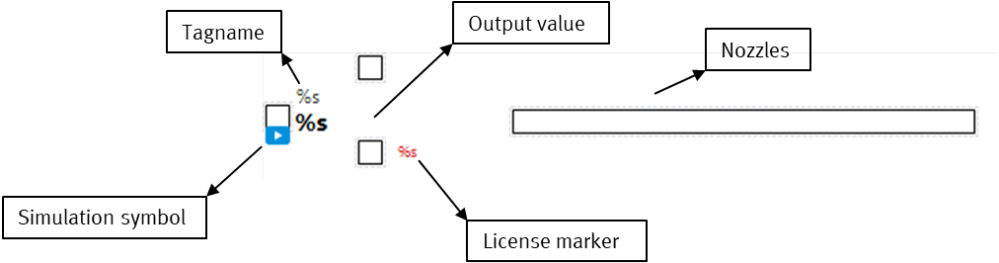


Fig. 39 Symbol hBinViewValue

7.2 Funktionsbaustein BinMon

7.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinMon erweitert den FB BinView um die Überwachung eines Alarmzustandes. Zusätzlich gibt es eine Diagnosefunktion, um ein flatterndes Signal zu erkennen.

7.2.2 Erweiterte Funktionen

7.2.2.1 Diagnosefunktion zur Erkennung eines flatterfreien Signals

Bei binären Eingängen kann es unter Umständen zu einem Flattern am Eingang der Steuerung kommen. Ein solches Flattern kann zu ungewolltem Verhalten der Steuerungsapplikation führen. Daher ist es sinnvoll ein solches Flattern zu diagnostizieren, um die Qualität des Signals zu kennen. Die Überwachung eines flatternden Signals kann eingeschaltet werden in dem der Eingang „VFlutEn“ am FB auf TRUE gesetzt wird. Zusätzlich müssen folgende Eingänge konfiguriert werden:

- VFlutTi: Überwachungszeitraum innerhalb dessen die Anzahl der aufgetretenen Flatterer gezählt werden. Der Wert wird in Sekunden angegeben und ist standardmäßig auf 60s eingestellt. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.
- VFlutCnt: maximale Anzahl von flatternden Signalen welche im Zeitraum „VFlutTi“ auftreten dürfen. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5 eingegeben so wird die Anzahl auf 5 gesetzt.
- FlutLen: minimale Länge eines flatterfreien Signals. Liegt ein Signal länger als diese Zeit an gilt es nicht als Flattern, sondern als Signal. Der Wert wird in Sekunden angegeben und ist standardmäßig auf 0.5s eingestellt. Werden Werte < 0.5 eingegeben so wird bei der Berechnung automatisch 0.5s angenommen.

Das Ergebnis wird am Ausgang „VFlutAct“ ausgegeben. Ist dieser Ausgang FALSE so ist das Signal flatterfrei.

7.2.2.2 Überwachung Alarmzustand

Der Ausgang „V“ kann den Wert TRUE oder FALSE annehmen. Wird bei Setzen eines dieser beiden Zustände ein Alarm benötigt, so kann diese Funktion über die Eingangsvariable „VAlmStateEn“ (= TRUE) aktiviert werden.

Über den Eingang am FB „VAlmState“ kann eingestellt werden, bei welchem Zustand ein Alarm ausgelöst wird. Damit der Alarm bei einem flatternden Signal nicht ausgelöst wird, muss der Alarmzustand eine gewisse Zeit anliegen, damit er als dieser identifiziert wird.

Dazu kann über den Eingang „VAlmTi“ eine Zeit angegeben werden. Erst wenn das Signal diese Zeit anliegt, wird es als Alarmzustand angesehen und der FB gibt einen Fehler am Ausgang aus, indem „VAlmStateAct“ auf TRUE gesetzt wird.

7.2.2.3 Zeit „VAlmTi“

Die Variable „VAlmTi“ wird in Sekunden angegeben und ist standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen < 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

7.2.2.4 Alarm quittieren

Ist der Ausgang nicht mehr im Alarmzustand, so wird der Ausgang „VALmStateAct wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („VALmAckOp“) oder über den Eingang „VALmAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 35 Alarmzustände im Symbol

Hinweis:

- Der Eingang „VALmAckAut“ ist flankengesteuert und reagiert nur auf steigende Flanken.

7.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSImEn (= TRUE)
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 36 BinMon spezifischer Worst Quality Code

7.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über **Applikation -> Add Object -> Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VFlutEn	BOOL	FALSE	Enables Fluttering Recognition (1: enabled 0: disabled)
VFlutTi	REAL	60	Period of an active signal before it will be recognized as flutter-free [s]
VFlutLen	REAL	0.5	Signal length of an active signal so that it is recognized as a signal and not as a fluttering.[s]
VFlutCnt	INT	0	Counts of the allowed fluttering signals in the defined period VFlutTi
VAlmStateEn	BOOL	FALSE	Enables the Alarm State Monitoring (1: enabled 0: disabled)
VAlmState	BOOL	FALSE	State for Alarm State Monitoring (1: TRUE is the alarm state 0: FALSE is the alarm state)
VAlmTi	REAL		Monitor Time to recognize the state as an alarm state [s]

Tab. 37 BinMon persistente Variablen

7.2.5 Symbole

7.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Alarm symbols	 : Alarm ist aktiv (höchste Priorität)	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „VSim“ wird der Ausgangswert eingegeben.	Erscheint wenn VSimEn = TRUE ist.
Output value	Ausgangswert „V“ als Text.	Je nach Wert wird Text von „State0“ oder „State1“ angezeigt.
Nozzles	Nozzles zur Verbindung von Measurement Lines (durchgezogene Linie) und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 38 BinMon Symbolbeschreibung

7.2.5.2 hBinMonValue

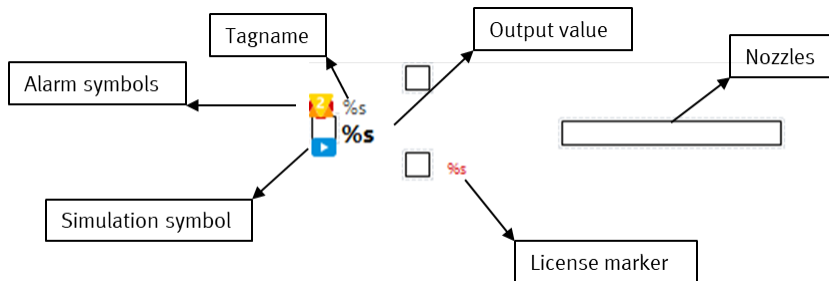


Fig. 40 Symbol hBinMonValue

7.2.6 Faceplates

7.2.6.1 Main

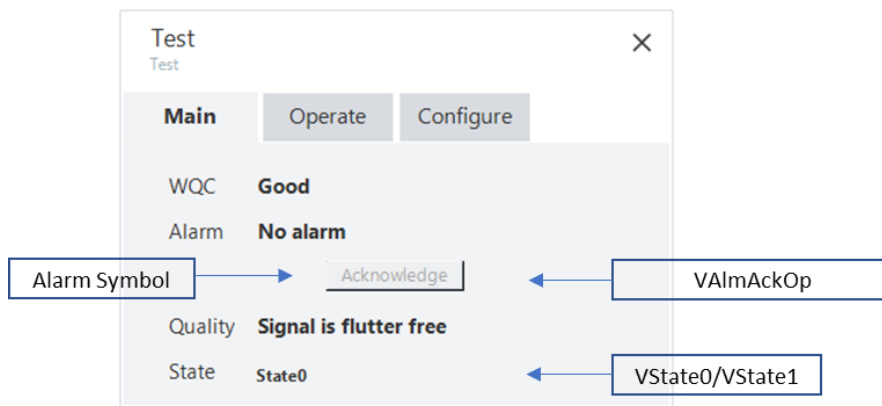


Fig. 41 BinMon Main Faceplate

7.2.6.2 Operate

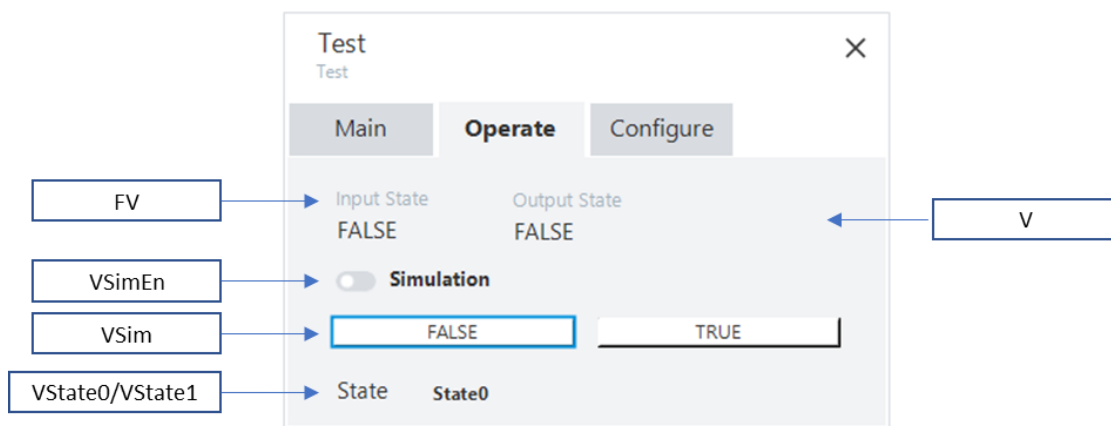


Fig. 42 BinMon Operate Faceplate

7.2.6.3 Configure

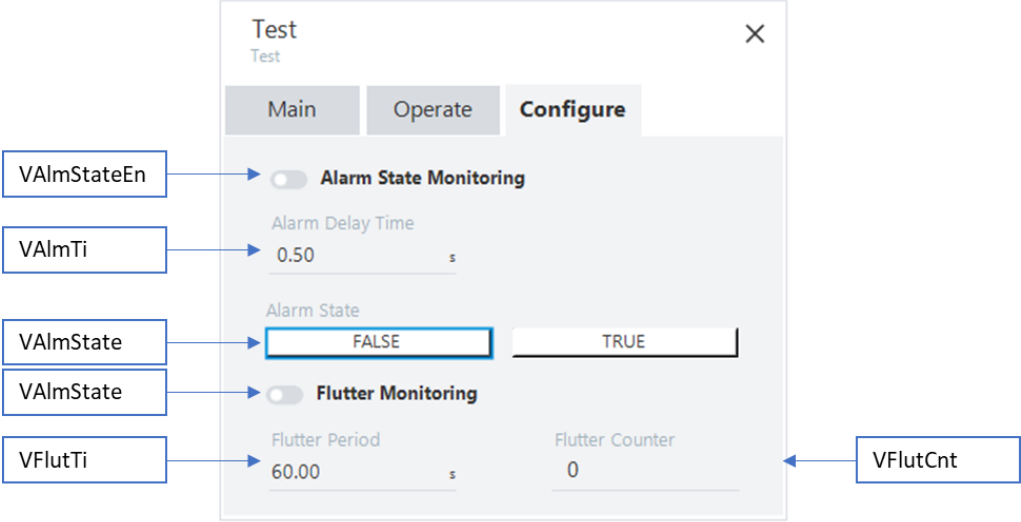


Fig. 43 BinMon Configure Faceplate

8 Binäroperationen

8.1 Funktionsbaustein BinMan

8.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinMan kann zum Übertragen eines binären Wertes (TRUE/FALSE) aus der Prozessfeldebene genutzt werden. Das Signal kann in der Applikation weiterverarbeitet oder an eine digitale Ausgangskarte ausgegeben werden.

8.1.2 Grundfunktionen

8.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

8.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 39 Worst Quality Code Enumeration

8.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

8.1.3 Funktionen

8.1.3.1 Eingangswert „VMan“

Der Eingangswert wird durch die Variable „VMan“ repräsentiert und wird als OPC UA Variable zur Verfügung gestellt. Der Eingangswert wird zur weiteren Verarbeitung an den Ausgang „VOut“ ausgegeben.

8.1.3.2 Statustext „VState0“ und „VState1“

Am FB stehen die Eingänge „VState0“ und „VState1“ als zwei String-Variable zur Verfügung. An diesen Eingängen können Texte für die beiden Zustände hinterlegt werden. Diese beschreiben zusätzlich die Zustände in Worten. So kann z.B. für den Wert TRUE der Eingang VState1 mit dem Text „Lampe an“ belegt werden.

Für jeden Statustext stehen 24 Zeichen zur Verfügung.

8.1.3.3 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den Wert von „VMan“ als Ausgang aus.

8.1.3.4 Feedbacksignal „VFbk“

Der Eingang „VFbkSignal“ kann direkt mit dem Feedbacksignal aus dem Feld verbunden werden. Dieser wird zur weiteren Verarbeitung direkt an den Ausgang „VFbk“ weitergeleitet.

8.1.3.5 Sicherheitszustand

Über den Eingang „VSafeState“ kann eine Sicherheitsposition angegeben werden. Im Falle eines Fehlers (siehe Lock-Funktionen) wird dieser Wert am Ausgang „VOut“ ausgegeben. Wird die Sicherheitsposition am „VOut“ ausgegeben, dann wird zusätzlich der Ausgang „SafePosAct“ auf TRUE gesetzt.

8.1.3.6 Lock-Funktionen

8.1.3.6.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

8.1.3.6.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird.

Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

8.1.3.6.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate (ResetOp) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf ansteigende Flanken.

8.1.4 FB-spezifischer Worst Quality Code

Nach der MTP-Spezifikation ist kein WQC in dieser Ausbaustufe vorgesehen.

8.1.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafeState	BOOL	FALSE	Safe State for Exception
OSLevel	BYTE	0	OS Level Variable

Tab. 40 BinMan persistente Variablen

8.1.6 Symbole

8.1.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Input value	„VMan“ wird ausgegeben.	Je nach Wert wird Text von „State0“ oder „State1“ angezeigt.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 41 BinMan Symbolbeschreibung

8.1.6.2 hBinManValue

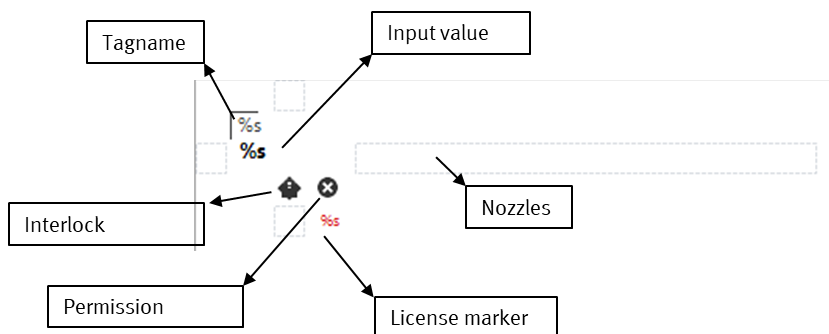


Fig. 44 Symbol hBinManValue

8.2 Funktionsbaustein BinManInt

8.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinManInt erweitert den FB BinMan um einen weiteren Eingangswert (VInt), so dass von zwei unterschiedlichen Quellen ein Eingangswert gesetzt werden kann.

8.2.2 Erweiterte Funktionen

8.2.2.1 Eingangswert „VMan“ und „VInt“

Bei dem FB BinMan stand der Eingang „VMan“ als OPC UA Variable zur Verfügung. Bei diesem FB gibt es einen zusätzlichen Eingangswert „VInt“ als Eingang am FB.

Über den Source Mode kann eingestellt werden, welcher Eingang an „VOut“ ausgegeben wird. Die Bedienung des Source Modes wird in einem extra Abschnitt erklärt.

8.2.2.2 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt je nach Einstellung des Source Modes den Wert von „VMan“ oder „VInt“ aus.

8.2.3 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 42 BinManInt spezifischer Worst Quality Code

8.2.4 Source Mode

8.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang SrcIntAct auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit SrcManAct auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

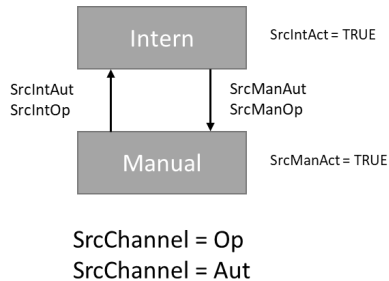


Fig. 45 Zustandsdiagramm Source Mode

8.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

8.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcManOp“, „SrcIntAut“ oder „SrcManAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt werden. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

8.2.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafeState	BOOL	FALSE	Safe State for Exception
OSLevel	BYTE	0	OS Level Variable

Tab. 43 BinManInt persistente Variablen

8.2.6 Symbole

8.2.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Input value	„VMan“ oder „VInt“ wird ausgegeben.	Je nach Wert wird Text von „State0“ oder „State1“ angezeigt.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 44 BinManInt Symbolbeschreibung

8.2.6.2 hBinManIntValue

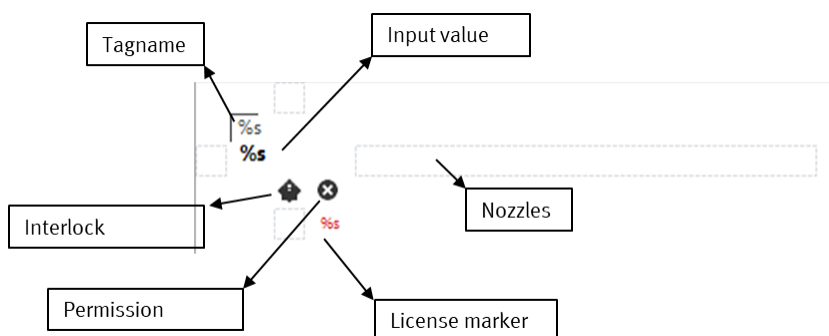


Fig. 46 Symbol hBinManIntValue

8.2.7 Faceplates

8.2.7.1 Main

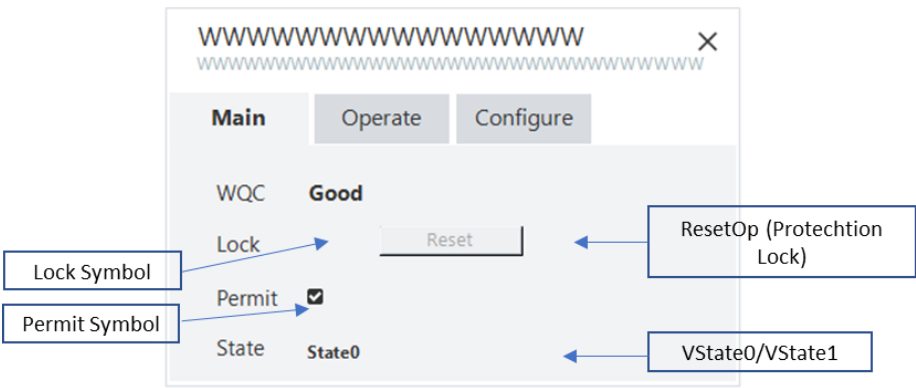


Fig. 47 BinManInt Main Faceplate

8.2.7.2 Operate

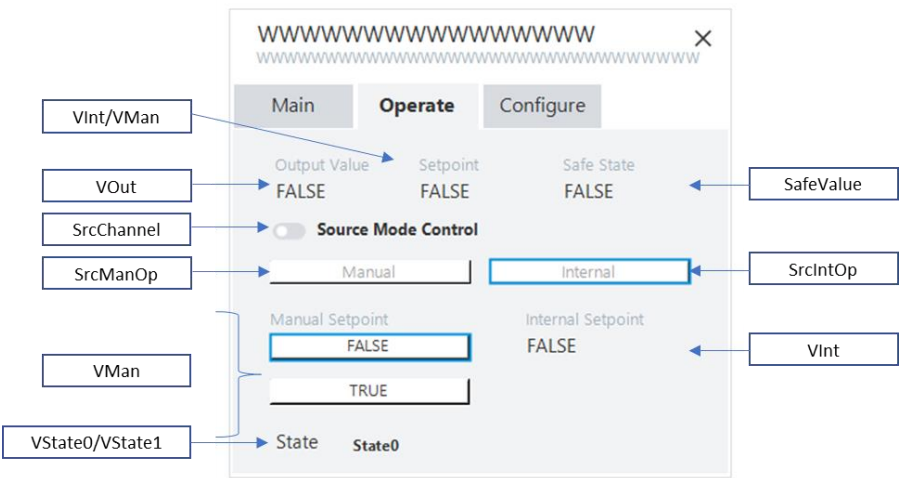


Fig. 48 BinManInt Operate Faceplate

8.2.7.3 Configure

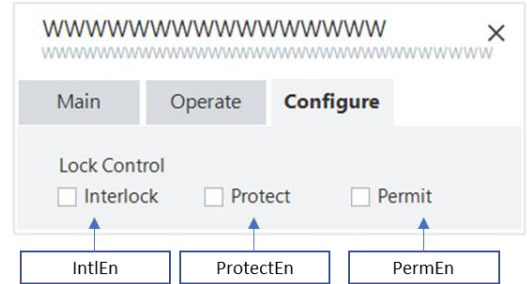


Fig. 49 BinManInt Configure Faceplate

9 Zeichenketteanzeige

9.1 Funktionsbaustein StrView

9.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) StringView kann zur Anzeige eines Strings in einem Visualisierungssystem verwendet werden.

9.1.1.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

9.1.1.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 45 Worst Quality Code Enumeration

9.1.2 Funktionen

9.1.2.1 Eingangswert „Text“

Über die Eingangsvariable „Text“ kann eine Zeichenfolge hinterlegt werden. Die Zeichenfolge kann direkt eingegeben oder über eine Variable hinterlegt werden. Es können maximal 24 Zeichen eingegeben werden. Die restlichen Zeichen werden in der Visualisierung abgeschnitten.

9.1.3 FB-spezifischer Worst Quality Code

Dieser FB hat keinen spezifischen WQC.

9.1.4 Persistente Variablen

Dieser FB hat keine persistenten Variablen.

9.1.5 Symbole

9.1.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Text	Text-Variable wird ausgegeben.	Max. 24 Zeichen werden dargestellt.
Nozzles	Nozzles zur Verbindung von und Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 46 StrView Symbolbeschreibung

9.1.5.2 hStrViewValue

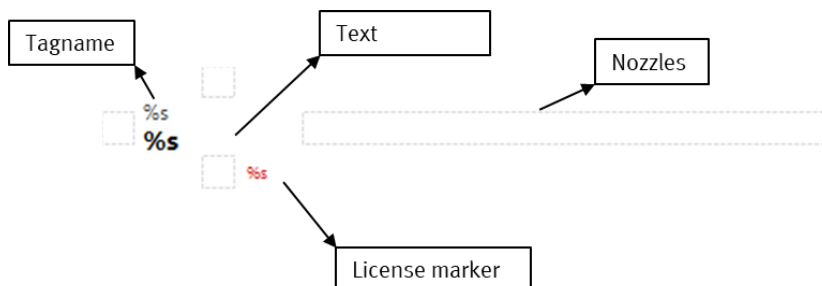


Fig. 50 Symbol hStrViewValue

9.1.6 Faceplate

Es gibt bei diesem FB kein Faceplate

10 Regelungen

10.1 Funktionsbaustein PIDCtrl

10.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) PIDCtrl stellt die Informationen für einen kontinuierlichen PID Regler bereit, der die typischen Größen (Prozesswert PV, Sollwert SP, Regelgröße MV) entsprechend verarbeitet. Die Schnittstelle beinhaltet für den Sollwert eine Umschaltung, um sowohl einen intern vorgegebenen als auch einen manuell vorgegebenen Sollwert zu verwenden. Hierbei kommt die Funktion Source Mode zum Einsatz. Der Regler selbst besitzt drei Zustände – Offline, Operator und Automatic. Im Zustand Operator, wird die Regelgröße MV über die Variable „MVMan“ durch den Operator vorgegeben. Im Zustand Automatic ist der Regelungsalgorithmus in Kraft. Dazu wird je nach Source Mode der interne oder manuelle Sollwert verarbeitet. Mit Hilfe der Min- und Max-Variablen werden Begrenzungsfunktionen für Sollwerte und Stellgröße dargestellt. Der Regelungsalgorithmus kann über die drei Variablen P, Ti und Td parametrisiert werden.

10.1.2 Grundfunktionen

10.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

10.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 47 Worst Quality Code Enumeration

10.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

10.1.3 Funktionen

10.1.3.1 Zustandsautomat – Operation Mode

Der PID-Regler wird in drei Zustände Offline, Operator und Automatic unterteilt. Die Zustände werden über den Operation Mode gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Im Zustand Offline, werden keine Werte verarbeitet. Der Ausgang „MV“ wird auf 0 gesetzt.

Im Zustand Operator kann der Operator über OPC UA oder über ein Faceplate den Ausgang „MV“ auf einen festen Wert setzen. Dieser Wert wird über die Variable „MVMan“ vorgegeben.

Im Zustand Automatic läuft der Regelalgorithmus (PID-Regler). Über den Source Mode kann eingestellt werden, ob „VMan“ oder „VInt“ als Setpoint für die Berechnung herangezogen wird.

10.1.3.2 Manipulated Value (MVMan)

Der manipulierte Wert „MVMan“ steht als OPC UA Variable oder über das Faceplate zur Verfügung. Dieser Wert wird im Zustand Operator verwendet. In diesem Zustand ist der Regelalgorithmus inaktiv und der eingegebene „MVMan“ wird an den Ausgang MV übergeben.

Die Eingänge am FB „MVMin“ und „MVMax“ begrenzen den „MVMan“. Unter-bzw. überschreitet „MVMan“ diese Grenzen so wird der jeweilige Grenzwert als Ausgang „MV“ gesetzt.

10.1.3.3 Prozess Value (PV oder simulierter Wert)

Der Prozesswert kann über den Eingang „PV“ in den FB zurückgeführt werden.

Ist kein Prozesswert verfügbar, kann über den Eingang „PVSIm“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „PVSImEn“ (=TRUE) aktiviert.

10.1.3.4 Parameter PID-Regler

Im Zustand Automatic wird über den PID- Regelungsalgorithmus der Ausgang „MV“ berechnet. Der PID-Regler setzt sich aus einem proportionalen Anteil (P), einem integralen Anteil (Ti) und einem differenzialen Anteil (Td) zusammen. Alle drei Werte müssen über die dafür vorgesehenen Eingänge am FB parametrisiert werden.

10.1.3.5 Sollwertvorgabe „SPMan“ und „SPInt“

Im Zustand Automatic kann für den Regelalgorithmus ein interner oder manueller Setpoint vorgegeben werden. Der interne Setpoint „SPInt“ steht als Eingang am FB zur Verfügung. Der manuelle Setpoint „SPMan“ kann über OPC UA oder über das Faceplate gesetzt werden. Welcher Setpoint für die Regelung verwendet wird, kann über den Source Mode eingestellt werden. Die Steuerung des Source Modes wird in einem extra Abschnitt erklärt. Die Eingänge am FB „SPIntMin“/„SPManMin“ und „SPIntMax“/„SPManMax“ begrenzen den dazugehörigen Setpoint. Unter-bzw. überschreitet der Setpoint seine Grenzwerte so wird der jeweilige Grenzwert als Setpoint gesetzt und zur weiteren Verarbeitung herangezogen.

10.1.3.6 Ausgangswert „MV“

Der Ausgangswert „MV“ wird im Zustand Operator durch die Eingangsvariable „MVMan“ vorgegeben. Im Zustand Automatic wird „MV“ durch den Regelalgorithmus berechnet. Zusätzlich hat der „WorkMode“ Einfluss auf den Ausgang. Im Automatic Mode wird „MV“ nicht auf die Grenzwerte „MVMin“ und „MVMax“ überprüft.

10.1.3.7 Einheit für „MVMan“, „MV“, „PV“, „SPMant“ und „SPInt“

Für den manipulierten Wert, sowie für den Prozesswert den Setpoint und den Ausgangswert „MV“ können über die Eingänge, welche mit Unit gekennzeichnet sind, am FB Einheiten angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Units werden nur zur Darstellung verwendet und haben keinen Einfluss auf die Umrechnung.

10.1.3.8 Überwachung der Min- und Maxwerte

Wie oben beschrieben begrenzen die Min – und Maxwerte die Eingangs- und Ausgangswert. Zusätzlich muss darauf geachtet werden, dass die minimalen Grenzwerte nicht größer sind als die dazugehörigen maximalen Grenzwerte. Andernfalls wird nicht garantiert, dass die Berechnungen und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

10.1.3.9 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Setpoints, des Ausgangswertes, oder des Prozesswertes, werden jeweils eine Min und eine Max Variable („*SclMin“ und „*SclMax“) zur Verfügung gestellt. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die entsprechenden Grenzwerte gesetzt.

10.1.3.10 Eingangsparameter „WorkMode“

Über den Eingang „WorkMode“ kann eingestellt werden, ob es sich bei der Regelung um eine Durchflussregelung oder um eine Füllstandsregelung handelt. Wird der Eingang „WorkMode“ auf TRUE gesetzt, so handelt es sich um eine Füllstandsregelung. Das bedeutet, dass bei einer Differenz zwischen Sollwert und Prozesswert „MV“ kleiner wird. Wird der Eingang „WorkMode“ auf FALSE gesetzt so handelt es sich um eine Durchflussregelung und bei einer Differenz zwischen Sollwert und Prozesswert wird „MV“ größer.

10.1.3.11 Reset des integralen Anteils (ResetAut/ResetOp)

Wird der Eingang „ResetAut“ am FB oder „ResetOp“ über OPC UA oder Faceplate ausgeführt, so wird der integrale Anteil im PID-Regler zurückgesetzt. Der Eingang „ResetAut“ ist flankengesteuert und reagiert nur auf steigende Flanken. Wird ein Überlauf des integralen Anteils, so wird dieser an dem Ausgang "IntegralOverflow" angezeigt.

10.1.4 Erweiterte Funktionen

10.1.4.1 Deviation Alarm

Der Deviation Alarm überwacht die Differenz zwischen Sollwert und Prozesswert. Über den Eingang „AlmDevEn“ (=TRUE) kann die Alarmfunktionalität aktiviert werden. Über den Eingang „AlmDevLim“ kann der Wert für die maximale Differenz zwischen Sollwert und Prozesswert eingestellt werden.

Wird die eingegebene Differenz überschritten, so muss diese Überschreitung eine gewisse Zeit anliegen damit ein Fehler ausgegeben wird. Diese Zeit kann über den Eingangswert „DevAlmTi“ eingestellt werden.

Ein Alarm wird über das zugehörige Ausgangsbit „AlmDevAct“ ausgegeben. Wird eine Visualisierung in CoDeSys verwendet so wird der entsprechende Alarm im Symbol angezeigt.

10.1.4.2 Zeit „AlmDevTi“

Die Variable „AlmDevTi“ wird in Sekunden angegeben und ist standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

10.1.4.3 Alarm quittieren

Liegt keine Grenzwertverletzung mehr vor, so wird der Ausgang „AlmDevAct“ auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („AlmDevAckOp“) oder über den Eingang „AlmDevAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Differenzverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 48 Alarmzustände im Symbol

Hinweis:

- Der Eingang „AlmDevAckAut“ ist flankengesteuert und reagiert nur auf steigende Flanken.

10.1.5 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	PVSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				MVMan < MVMin
				MVMan > MVMax
				SPMan < SPManMin ⇒ SPMan = SPManMin
				SPMan > SPManMax ⇒ SPMan = SPManMax
				SPInt < SPIntMin ⇒ SPInt = SPIntMin
				SPInt > SPIntMax ⇒ SPInt = SPIntMax
				SPIntMin > SPIntMax
				SPManMin > SPManMax
				MVMin > MVMax
				PVScIMax < PVScImin ⇒ PVScIMax = SPIntMax ⇒ PVScImin = SPIntMin
				SPScIMax < SPScImin ⇒ SPScIMax = SPIntMax ⇒ SPScImin = SPIntMin
				MVScIMax < MVScImin ⇒ MVScIMax = MVMax ⇒ MVScImin = MVMin
				SpScIMax < SPIntMin oder SpScIMax > SPIntMax ⇒ SpScIMax = SPIntMax
				SpScImin < SPIntMin oder SpScImin > SPIntMax ⇒ SpScImin = SPIntMin
				MVScIMax < MVMin oder MVScIMax > MVMax ⇒ MVScIMax = MVMax
				MVScImin < MVMin oder MVScImin > MVMax ⇒ MVScImin = MVMin
				PVScIMax – PVScImin = 0 ⇒ PVScIMax = SPIntMax ⇒ PVScImin = SPIntMin
				SPScIMax – SPScImin = 0 ⇒ SPScIMax = SPIntMax ⇒ SPScImin = SPIntMin
				MVScIMax – MVScImin = 0 ⇒ MVScIMax = MVMax

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
				⇒ MVScMin = MVMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 49 PIDCtrl spezifischer Worst Quality Code

10.1.6 Source Mode

10.1.6.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit „SrcManAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

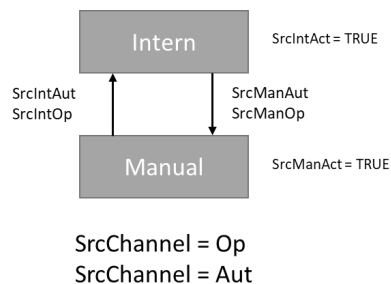


Fig. 51 Zustandsdiagramm Source Mode

10.1.6.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

10.1.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcManOp“, „SrcIntAut“ oder „SrcManAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt werden. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

10.1.7 Operation Mode

10.1.7.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline-Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

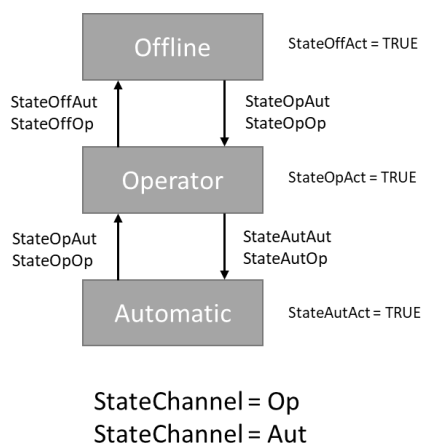


Fig. 52 Zustandsdiagramm Operation Mode

10.1.7.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline-Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline-Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline-Mode. Da vom Automatic Mode nicht in den Offline-Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

10.1.7.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline-Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline-Mode. Da vom Automatic Mode nicht in den Offline-Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

10.1.8 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
P	REAL	0.5	Proportional Parameter
Ti	REAL	0.5	Integration Parameter
Td	REAL	0	Derivation Parameter
WorkMode	BOOL	FALSE	Control Type Parameter (1: Level Control-MV decrease 0: Flow Control - MV increase)
AlmDevEn	BOOL	FALSE	Enables Deviation Alarm (1: enabled 0: disabled)
AlmDevLim	REAL	0	Limit Value for Deviation Alarm
AlmDevTi	REAL	0.5	Monitor Time for Deviation Alarm [s]
OSLevel	BYTE	0	OS Level Variable

Tab. 50 PIDCtrl persistente Variablen

10.1.9 Symbole

10.1.9.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Alarm symbols	 : Alarm ist aktiv (höchste Priorität)	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Simulation symbol	Zeigt an, wenn der Ausgangswert simuliert wird. Über „PVSIm“ wird der Ausgangswert eingegeben.	Erscheint wenn PVSImEn = TRUE ist.
Process Value	„PV“ oder „PVSIm“ werden dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
PV Unit	Einheit „PVUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Setpoint	„Splnt“ oder „SpMan“ wird dargestellt.	Min- und Maxwerte werden berücksichtigt.
SP Unit	Einheit „SpUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
Bargraph	Stellt den Ausgangswert „MV“ grafisch dar. „MVSclMin“ und „MVSclMax“ skalieren.	Skalierungswerte müssen richtig eingestellt sein sonst wird der Bargraph fehlerhaft dargestellt. („MVSclMin“ < „MVSclMax“).
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 51 PIDCtrl Symbolbeschreibung

10.1.9.2 hPIDCtrlDefault

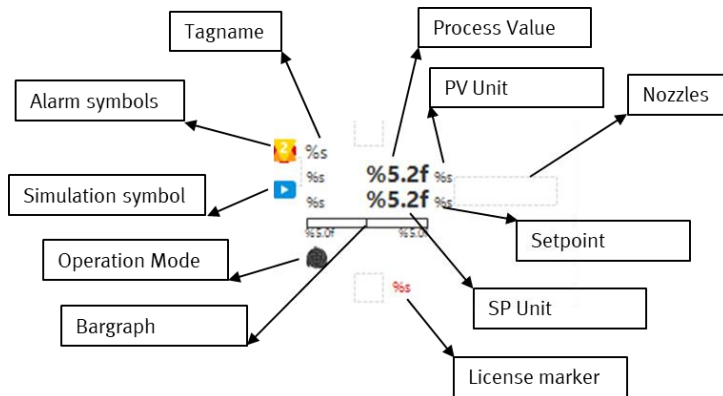


Fig. 53 Symbol hPIDCtrlDefault

10.1.10 Faceplates

10.1.10.1 Main

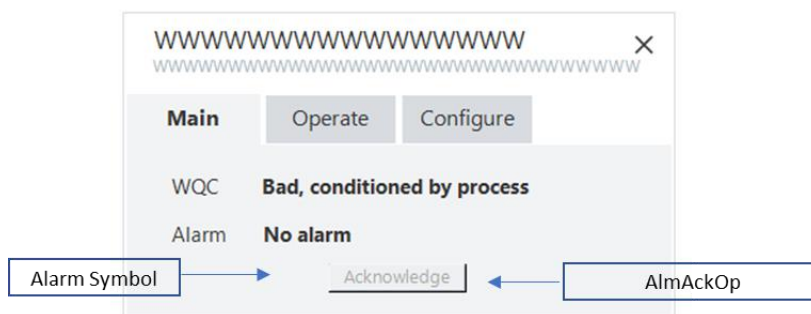


Fig. 54 PIDCtrl Main Faceplate

10.1.10.2 Operate

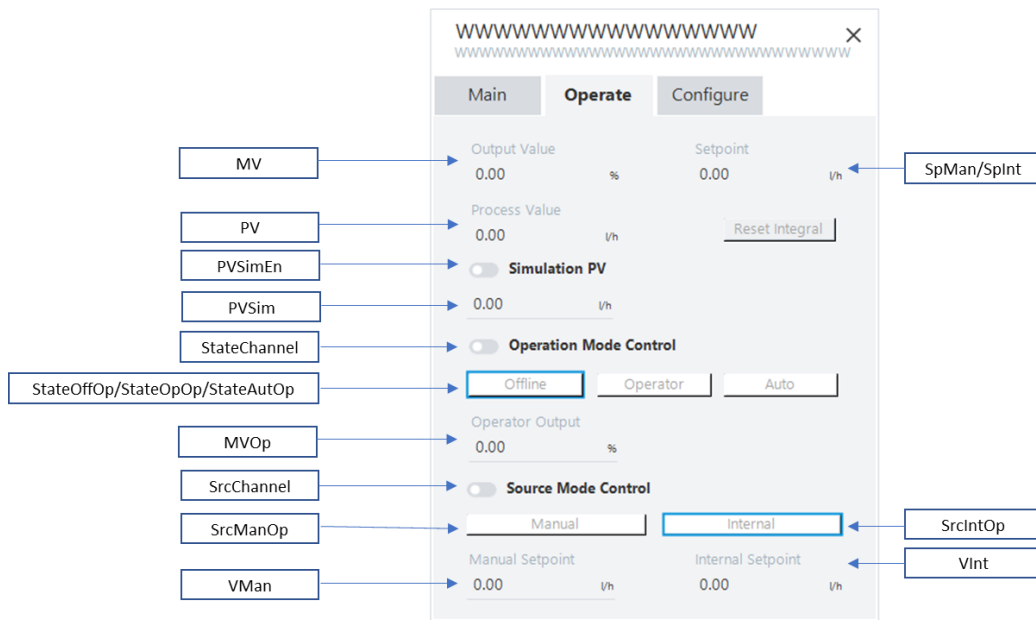


Fig. 55 PIDCtrl Operate Faceplate

10.1.10.3 Configure

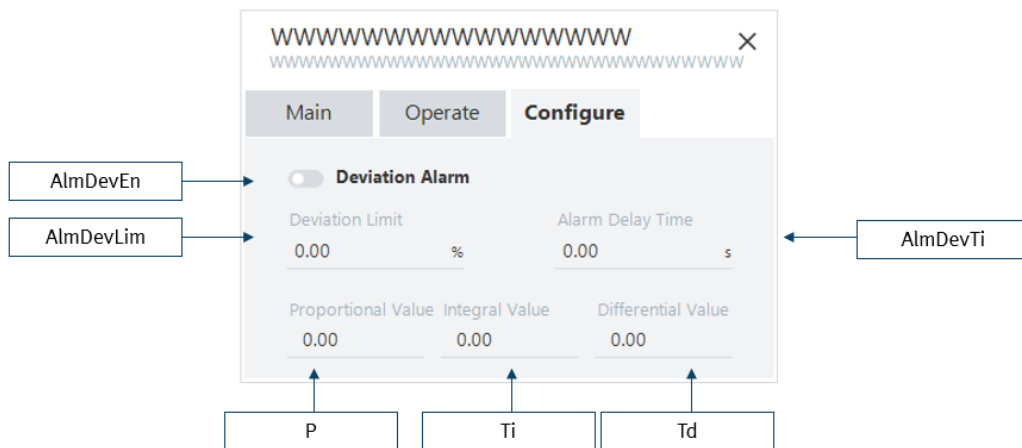


Fig. 56 PIDCtrl Configure Faceplate

11 Bistabile Ventile

11.1 Funktionsbautein BinVlv

11.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinVlv bietet die Möglichkeit ein bistabiles Ventil (einfach- oder doppelwirkend) zu steuern. Steuerungskommandos können entweder direkt am FB (Aut-Variablen) oder über ein Faceplate/OPC UA (Op-Variablen) gesetzt werden. Zusätzlich stehen Feedbacksignale zur Verfügung, welchen den tatsächlichen Ventilzustand anzeigen.

11.1.2 Grundfunktionen

11.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

11.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 52 Worst Quality Code Enumeration

11.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

11.1.3 Funktionen

11.1.3.1 Ventilsteuerung

Das bistabile Ventil kann entweder geöffnet oder geschlossen werden. Zur Steuerung des Ventils stehen Eingänge am FB (OpenAut und CloseAut) und Variablen über OPC UA/Faceplate (OpenOp und CloseOp) bereit. Das entsprechende Steuerkommando wird im FB verarbeitet und an den Ausgang „Ctrl“ ausgegeben. Damit ein doppelwirkendes Ventil sowie Normaly Close und Normaly Open Ventile angesteuert werden können, wird am Ausgang „CtrlInv“ das invertierte Signal zu „Ctrl“ ausgegeben.

Ein Steuerbefehl wird abgegeben indem die oben genannten Variablen je nach Befehl auf TRUE gesetzt werden. Die Eingangsvariablen „OpenAut“ und „CloseAut“ sind flankengesteuert und reagieren auf ansteigende Flanken. „OpenOp“ und „CloseOp“ werden nach der Registrierung vom FB wieder auf FALSE gesetzt.

In der folgenden Tabelle ist aufgeführt wie sich die Ausgänge in Abhängigkeit der Sicherheitsposition und den Steuerkommandos verhalten.

Steuerkommando	Output
Open	Ctrl = 1 CtrlInv = 0
Close	Ctrl = 0 CtrlInv = 1
Open und Close gleichzeitig	Ctrl = SafePos CtrlInv = !SafePos

Tab. 53 Steuerkommandos für Ventile

Der FB kann in drei Zustände Offline, Operator und Automatic betrieben werden. Die Zustände werden über den Operation Mode gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Folgende Steuerbefehle und Ausgänge werden im jeweiligen Zustand verarbeitet und gesetzt:

- Offline Mode : Ctrl = SafePos, CtrlInv = !SafePos
- Operator Mode: OpenOp und CloseOp steuern Ctrl und CtrlInv (über Faceplate änderbar)
- Automatic Mode: OpenAut und CloseAut steuern Ctrl und CtrlInv (Eingänge am FB)

11.1.3.2 Sicherheitsposition „SafePos“ und „SafePosEn“

Über den Eingang „SafePos“ kann eingestellt werden, ob die Ruhe-bzw.- Sicherheitsposition offen (SafePos = TRUE) oder geschlossen (SafePos = FALSE) ist.

Über den Eingang „SafePosEn“ kann eingestellt werden, ob das Ventil einen Sicherheitszustand hat. Falls es keinen Sicherheitszustand gibt, wird immer der letzte Zustand bei einem Fehler gehalten.

11.1.3.3 Feedbacksignale (simuliert, tatsächlicher Ventilzustand)

Feedbacksignale werden automatisch vom FB selbst simuliert, wenn die Eingänge „OpenFbkCalc“ und/oder „CloseFbkCalc“ auf TRUE gesetzt sind.

Falls keine Simulation gewünscht ist, können die Eingänge „OpenFbkSignal“ und/oder „CloseFbkSignal“ mit den

tatsächlichen Feedbacksignalen des Ventils verbunden werden. Zusätzlich müssen „OpenFbkCalc“ und „CloseFbkCalc“ auf FALSE gesetzt werden. Wird eine Endlage erreicht, so wird der entsprechende Ausgang am FB (OpenFbk oder CloseFbk) auf TRUE gesetzt. Als Standardeinstellung werden die richtigen Feedbacksignale über „OpenFbkSignal“ und/oder „CloseFbkSignal“ verarbeitet.

11.1.3.4 Lock-Funktionen

11.1.3.4.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

11.1.3.4.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

11.1.3.4.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate (ResetOp) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf ansteigende Flanken.

11.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 54 BinVlv spezifischer Worst Quality Code

11.1.5 Operation Mode

11.1.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline-Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

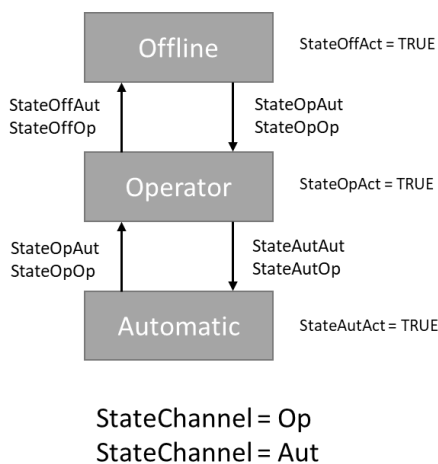


Fig. 57 Zustandsmaschine Operation Mode

11.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline-Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline-Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline-Mode. Da vom Automatic Mode nicht in den Offline-Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

11.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline-Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline-Mode. Da vom Automatic Mode nicht in den Offline-Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

11.1.6 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePosEn	BOOL	FALSE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePos	BOOL	TRUE	Safe Position (1: Open 0: Close)
OSLevel	BYTE	0	OS Level variable

Tab. 55 BinVlv persistente Variablen

11.1.7 Symbole

11.1.7.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Valve symbol	Grau: „CloseFbk“ ist aktiv Blau: „OpenFbk“ ist aktiv	
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	 : kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles) und Pipes (blaue Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 56 BinVlv Symbolbeschreibung

11.1.7.2 hBinVlvValue

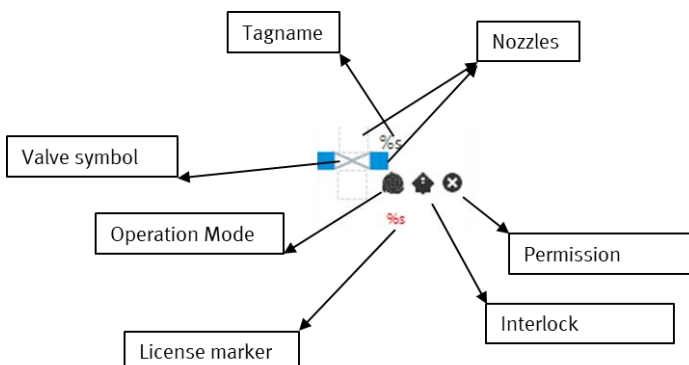


Fig. 58 Symbol hBinVlvValue

11.1.7.3 vBinVlvBalue

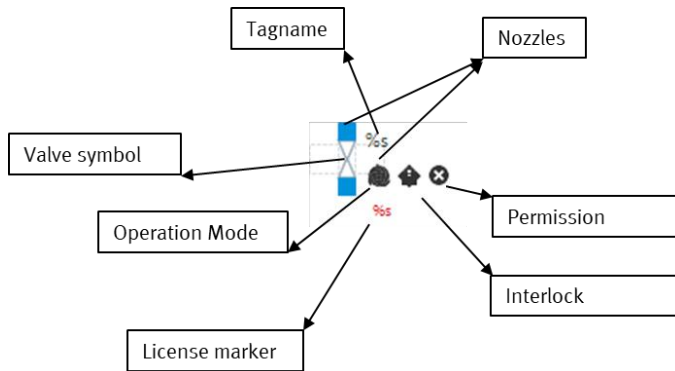


Fig. 59 Symbol vBinVlvValue

11.1.7.4 dldrThreeWayVlvValue, drdlThreeWayVlvValue, ldluThreeWayVlvValue, luldThreeWayVlvValue, rdruThreeWayVlvValue, rurdThreeWayVlvValue, ulurThreeWayVlvValue, urulThreeWayVlvValue

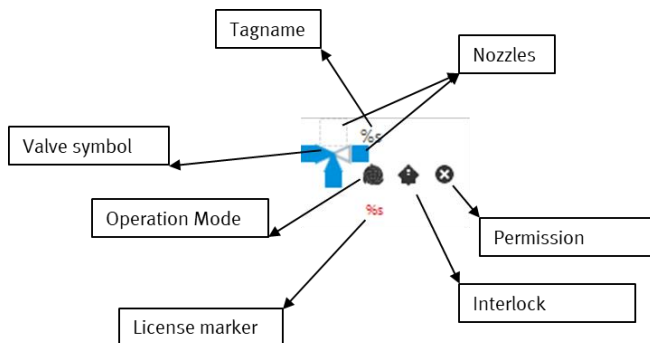


Fig. 60 Symbole 3-Wege-Ventile

11.2 Funktionsbaustein MonBinVlv

11.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) MonBinVlv erweitert den FB BinVlv um das Feedback Monitoring von dynamischen und statischen Fehlern.

11.2.2 Erweiterte Funktionen

11.2.2.1 Feedback Monitoring aktivieren

Das Feedback-Monitoring (statisch und dynamisch) kann über den Eingang am FB „MonEn“ (=TRUE) aktiviert werden. Ist „MonEn“ auf FALSE gesetzt oder wird im laufenden Betrieb auf FALSE gesetzt so werden alle Fehlerausgänge zurückgesetzt. Auch in der Visualisierung verschwindet das Alarm-Symbol und der Fehler muss nicht quittiert werden.

11.2.2.2 Eingang „MonSafePos“

Über den Eingang am FB „MonSafePos“ kann eingestellt werden, ob bei einem Fehler das Ventil die Sicherheitsposition anfährt (MonSafePos = TRUE) oder, ob der aktuelle Zustand gehalten werden soll (MonSafePos = FALSE). Bei beiden Varianten wird das Ausgangsbit SafePosAct auf TRUE gesetzt, damit der Ausnahmezustand erkannt wird. Als Standardeinstellung wird bei einem Fehler die Sicherheitsposition angefahren.

Soll nach einem Fehler der letzte Zustand gehalten werden oder befindet sich das Ventil schon im Sicherheitszustand, wird ein Fehler erst beseitigt, wenn die entsprechenden Feedbacksignale die richtigen Werte liefern.

Wird nach einem Fehler das Ventil in den Sicherheitszustand gefahren, wird der Fehler erst behoben, wenn nochmals in den Zustand, wo der Fehler aufgetreten ist, verfahren wird und die Feedbacksignale die richtigen Werte liefern.

BSP: Das Ventil ist im Zustand Close und soll geöffnet werden. Nach dem Öffnen bleibt „OpenFbk“ auf FALSE. So wird das Ventil wieder in seinen Sicherheitszustand (hier Close) verfahren. Damit der Fehler verschwindet, muss das Ventil wieder geöffnet werden und „OpenFbk“ muss TRUE werden. Erst dann werden die Fehlerausgänge („StatErrAct“ und „DynErrAct“) wieder auf FALSE gesetzt.

11.2.2.3 Statische Fehler

Ein statischer Fehler liegt vor, wenn sich das Feedback-Signal ändert, ohne dass eine Änderung der Ansteuerung erfolgt ist.

Folgende statische Fehler können auftreten:

- „OpenFbk“ und „CloseFbk“ sind beide TRUE
- Zustand ist Open, „OpenFbk“ wird FALSE
- Zustand ist Close, „CloseFbk“ wird FALSE

Damit eine Störung nicht als verlorenes bzw. falsches Signal gewertet wird, muss dieser Fehlerzustand eine bestimmte Zeit anliegen damit es als Fehler gewertet wird. Diese Zeit kann über den Eingang am FB „MonStatTi“ eingestellt werden. Erst wenn diese Zeit abgelaufen ist und der Fehlerzustand immer noch anliegt, wird der Fehler erkannt und der Ausgang „StatErrAct“ wird auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

11.2.2.4 Dynamische Fehler

Ein dynamischer Fehler liegt vor, wenn sich das Feedback-Signal nicht ändert, obwohl eine Änderung der Ansteuerung erfolgt ist.

Folgende dynamische Fehler können auftreten:

- Zustand soll sich von Open nach Close ändern- „CloseFbk“ bleibt auf FALSE
- Zustand soll sich von Close nach Open ändern- „OpenFbk“ bleibt auf FALSE

Damit das Feedbacksignal nicht sofort erwartet wird, da ein Ventil Zeit zum Schließen und Öffnen benötigt kann über den Eingang am FB „MonDynTi“ eine Zeit eingestellt werden. Wird in dieser Zeit nicht das richtige Feedbacksignal erkannt wird der Ausgang „DynErrAct“ auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

11.2.2.5 Alarm Priorität

Folgende Priorität wird bei der Alarmanzeige in der Visualisierung eingehalten:

- Failed to open (höchste Priorität)
- Failed to close
- Both Feedbacks active
- Open Feedback lost
- Close Feedback lost
- No alarm

11.2.2.6 Zeit „MonStatTi“ und „MonDynTi“

Die Variablen „MonStatTi“ und „MonDynTi“ werden in Sekunden angegeben und sind standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

11.2.2.7 Alarm quittieren

Liegt kein Fehler mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („MonAckOp“) oder über den Eingang „MonAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 57 Zustände Alarmsymbol

Hinweis:

- Der Eingang „MonAckAut“ ist flankengesteuert und reagiert nur auf ansteigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarme gleichzeitig quittiert.
- Es können auch mehrere Fehler gleichzeitig anliegen – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

11.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 58 MonBinVlv spezifischer Worst Quality Code

11.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePosEn	BOOL	TRUE	Safe Position (1: Open 0: Close)
OSLevel	BYTE	0	OS Level variable
MonEn	BOOL	FALSE	Enables dynamic and static Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s], only positive values

Variable	Type	Default	Description
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s], only positive values

Tab. 59 MonBinVlv persistente Variablen

11.2.5 Symbole

11.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Valve symbol	Grau: „CloseFbk“ ist aktiv Blau: „OpenFbk“ ist aktiv	
Alarm symbols	 : Alarm ist aktiv	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles) und Pipes (blaue Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 60 MonBinVlv Symbolbeschreibung

11.2.5.2 hMonBinVlvValue

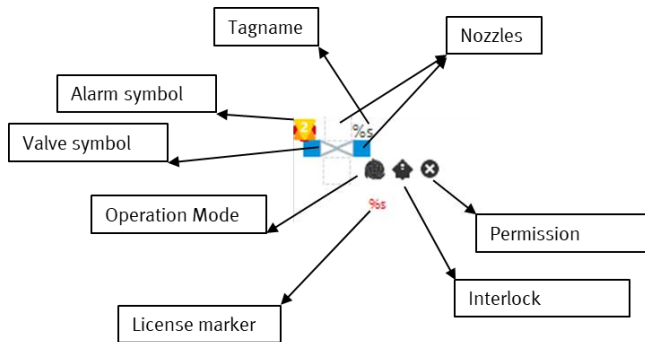


Fig. 61 Symbol hMonBinVlvValue

11.2.5.3 vMonBinVlvValue

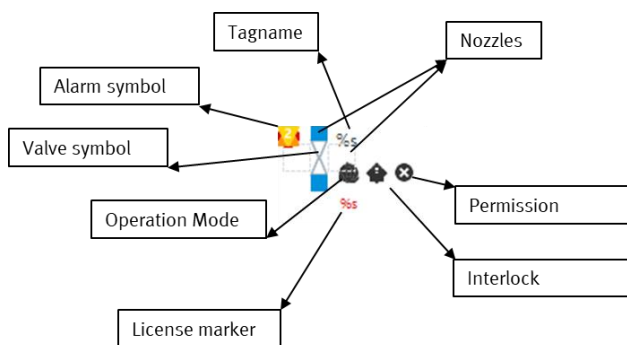


Fig. 62 Symbol vMonBinVlvValue

11.2.5.4 dldrMonThreeWayVlvValue, drdlMonThreeWayVlvValue, ldluMonThreeWayVlvValue, luldMonThreeWayVlvValue, rdruMonThreeWayVlvValue, rurdMonThreeWayVlvValue, ulurMonThreeWayVlvValue, urulMonThreeWayVlvValue

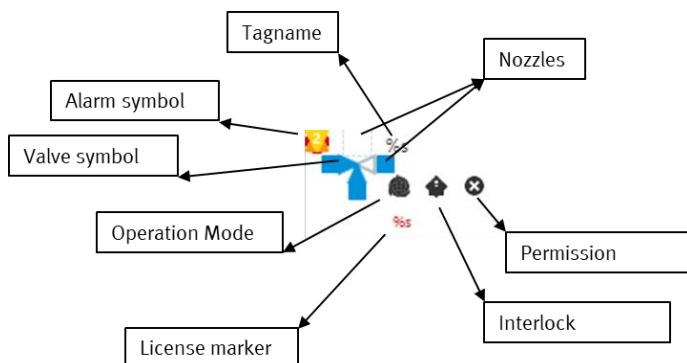


Fig. 63 Symbol Mon-3-Wege-Ventile

11.2.6 Faceplates

11.2.6.1 Main

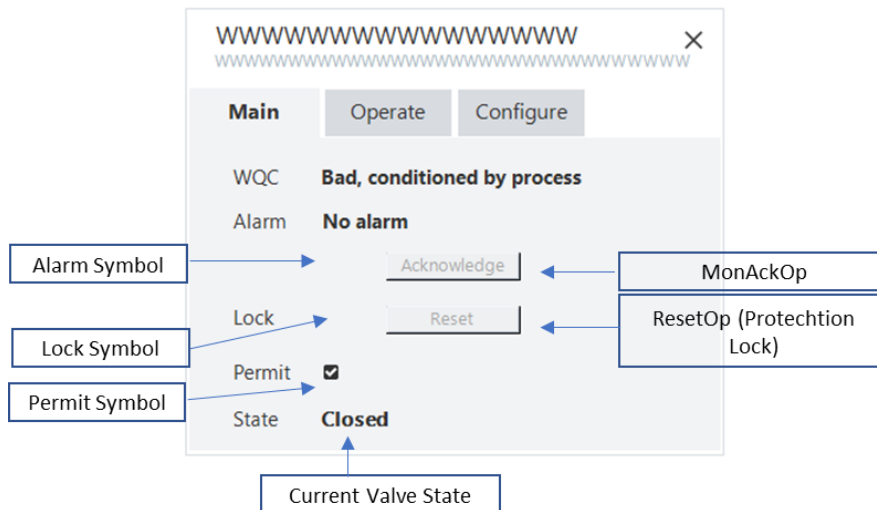


Fig. 64 MonBinVlv Main Faceplate

11.2.6.2 Operate

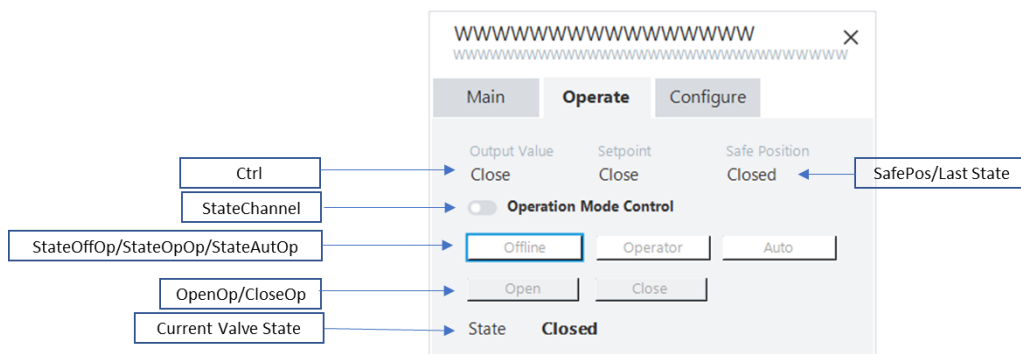


Fig. 65 MonBinVlv Operate Faceplate

11.2.6.3 Configure

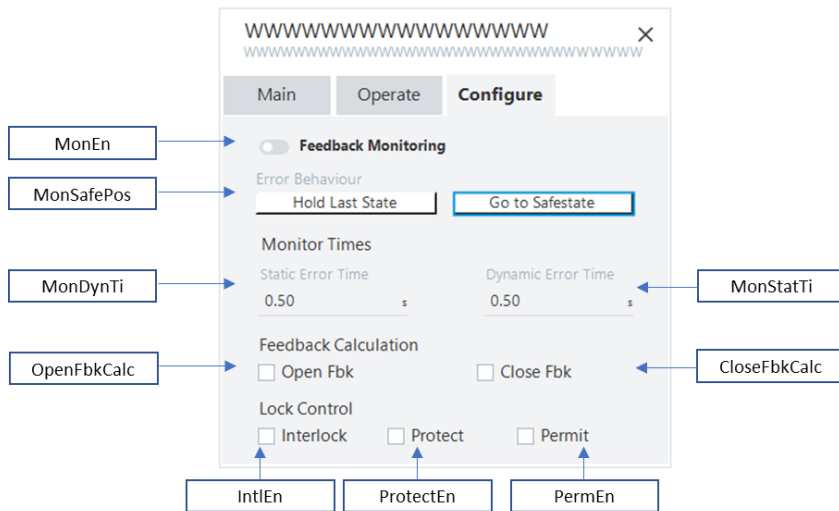


Fig. 66 MonBinVlv Configure Faceplate

12 Regelbare Ventile

12.1 Funktionsbaustein AnaVlv

12.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaVlv bietet die Möglichkeit ein Ventil zwischen einem Min- und Maxwert frei zu positionieren. Eine Sollposition kann über eine interne oder manuelle Quelle vorgegeben werden. Zusätzlich stehen Feedbacksignale zur Verfügung, welchen den tatsächlichen Ventilzustand anzeigen.

12.1.2 Grundfunktionen

12.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

12.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 61 Worst Quality Code Enumeration

12.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

12.1.3 Funktionen

12.1.3.1 Sollwertvorgabe „PosMan“ und „PosInt“

Der Sollwert für die Position kann über zwei Quellen angegeben werden. Über den Eingang „PosInt“ kann direkt am FB der Sollwert vorgegeben werden. Soll der Sollwert über OPC UA oder das Faceplate eingestellt werden, so wird das Eingabefeld „Manual Setpoint“ genutzt. Welcher der beiden Eingänge gerade für die Positionierung herangezogen wird steuert der Source Mode. Bei der Initialisierung ist die interne Quelle und somit „PosInt“ aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erklärt.

12.1.3.2 Begrenzung Sollwert

Über die Eingänge am FB „PosMin“ und „PosMax“ können Grenzwerte für die Position eingegeben werden. Ist der angegebene Sollwert („PosInt“ bzw. „PosMan“) größer „PosMax“ oder kleiner „PosMin“ so wird der Sollwert auf den entsprechenden Grenzwert begrenzt. Zusätzlich muss darauf geachtet werden, dass „PosMin“ nicht größer ist als „PosMax“. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

12.1.3.3 Sicherheitsposition „SafePos“ und „SafePosEn“

Über den Eingang „SafePos“ kann eingestellt werden, ob die Ruhe-bzw.- Sicherheitsposition die minimale Position „PosMin“ (SafePos = FALSE) oder die maximale Position „PosMax“ (SafePos = TRUE) ist.

Über den Eingang „SafePosEn“ kann eingestellt werden, ob das Ventil einen Sicherheitszustand hat. Falls es keinen Sicherheitszustand gibt, wird immer der letzte Zustand bei einem Fehler gehalten.

12.1.3.4 Ventilsteuerung

Das regelbare Ventil kann zwischen „PosMin“ und „PosMax“ frei positioniert werden. Zur Steuerung des Ventils stehen Eingänge am FB („OpenAut“ und „CloseAut“) und Variablen über OPC UA/Faceplate („OpenOp“ und „CloseOp“) bereit. Das entsprechende Steuerkommando wird im FB mit dem Sollwert „PosInt“ oder „PosMan“ verarbeitet und an den Ausgang „Pos“ ausgegeben.

Ein Steuerbefehl wird abgegeben indem die oben genannten Variablen je nach Befehl auf TRUE gesetzt werden. Die Eingangsvariablen „OpenAut“ und „CloseAut“ sind flankengesteuert und reagieren auf ansteigende Flanken. „OpenOp“ und „CloseOp“ werden nach der Registrierung vom FB wieder auf FALSE gesetzt.

In der folgenden Tabelle ist aufgeführt wie sich der Ausgang in Abhängigkeit der Sicherheitsposition und den Steuerkommandos verhalten.

Steuerkommando	Ausgänge
Open	Pos = PosInt/PosMan
Close	Pos = PosMin
Open und Close gleichzeitig	Pos = SafePos

Tab. 62 Steuerkommandos Ventile

Der FB kann in drei Zustände Offline, Operator und Automatic betrieben werden. Die Zustände werden über den Operation Mode gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Folgende Steuerbefehle und Ausgänge werden im jeweiligen Zustand verarbeitet und gesetzt:

- Offline Mode: Pos = SafePos
- Operator Mode: „OpenOp“ und „CloseOp“ steuern „Pos“, „SpMan“/„Splnt“ sind gültig (über Faceplate änderbar)
- Automatic Mode: „OpenAut“ und „CloseAut“ steuern „Pos“, „SpMan“/„Splnt“ sind gültig. Bei OpenAut wird Sollwert dauerhaft verfolgt

12.1.3.5 Einheiten für Position

Für die Position kann über den Eingang am FB „PosUnit“ eine Einheit angegeben werden. Um eine Einheit anzugeben wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

12.1.3.6 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung der Position, werden jeweils eine Min und eine Max Variable („PosSclMin“ und „PosSclMax“) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Min- und Maxwerte der Position begrenzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

12.1.3.7 Readback

Der Ausgang „PosRbk“ gibt den aktiven Eingangswert („PosMan“ oder „PosInt“) ohne Berücksichtigung der Grenzwerte „PosMin“ und „PosMax“ aus.

12.1.3.8 Feedbacksignale (simuliert, tatsächlicher Ventilzustand)

Feedbacksignale werden automatisch vom FB selbst simuliert, wenn die Eingänge „OpenFbkCalc“ und/oder „CloseFbkCalc“ auf TRUE gesetzt sind.

Falls keine Simulation gewünscht ist, können die Eingänge „OpenFbkSignal“ und/oder „CloseFbkSignal“ mit den tatsächlichen Feedbacksignalen des Ventils verbunden werden. Zusätzlich müssen „OpenFbkCalc“ und „CloseFbkCalc“ auf FALSE gesetzt werden. Wird eine Endlage erreicht, so wird der entsprechende Ausgang („OpenFbk“ oder „CloseFbk“) auf TRUE gesetzt. Als Standardeinstellung werden die richtigen Feedbacksignale über „OpenFbkSignal“ und/oder „CloseFbkSignal“ verarbeitet.

Ebenfalls kann die Positionsrückmeldung simuliert werden. Dazu muss der Eingang am FB „PosFbkCalc“ auf TRUE gesetzt werden. Die berechnete Position wird am Ausgang „PosFbk“ ausgegeben. Wird der Eingang „PosFbkCalc“ auf FALSE gesetzt, so kann über den Eingang am FB „PosFbkSignal“ die Rückmeldung vom Ventil direkt übertragen und an den Ausgang „PosFbk“ ausgegeben werden.

12.1.3.9 Lock-Funktionen

12.1.3.9.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

12.1.3.9.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

12.1.3.9.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate (ResetOp) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf ansteigende Flanken.

12.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				Pos < PosMin ⇒ Pos = PosMin
				Pos > PosMax ⇒ Pos = PosMax
				PosMin > PosMax
				PosSclMax < PosSclMin ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosSclMax
				PosSclMin < PosMin oder PosSclMin > PosMax ⇒ PosSclMin = PosMin
				PosSclMax < PosMin oder PosSclMax > PosMax ⇒ PosSclMax = PosMax
				PosSclMax – PosSclMin = 0 ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 63 AnaVlv spezifischer Worst Quality Code

12.1.5 Source Mode

12.1.5.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit „SrcManAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

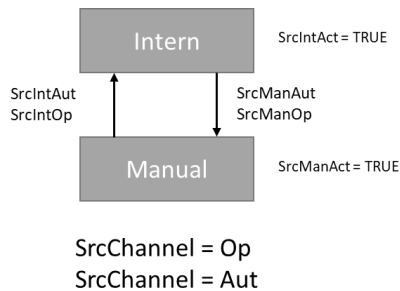


Fig. 67 Zustandsdiagramm Source Mode

12.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

12.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcManOp“, „SrcIntAut“ oder „SrcManAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt werden. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

12.1.6 Operation Mode

12.1.6.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline-Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

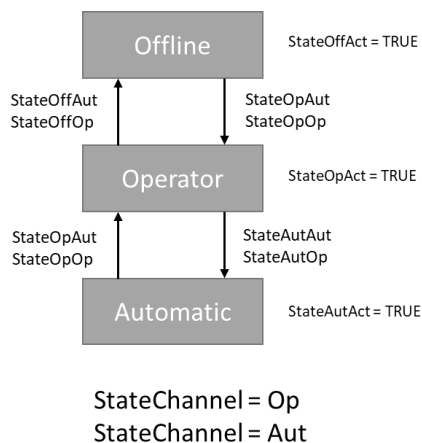


Fig. 68 Zustandsdiagramm Operation Mode

12.1.6.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline-Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline-Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline-Mode. Da vom Automatic Mode nicht in den Offline-Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

12.1.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline-Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline-Mode. Da vom Automatic Mode nicht in den Offline-Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

12.1.7 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePosEn	BOOL	TRUE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePos	BOOL	FALSE	Safe position (1: PosMax 0: PosMin)
OSLevel	BYTE	0	OS Level Variable

Tab. 64 AnaVlv persistente Variablen

12.1.8 Symbole

12.1.8.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Valve symbol	Grau: „CloseFbk“ ist aktiv Blau: „OpenFbk“ ist aktiv	
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Position setpoint Bargraph	„PosInt“ oder „PosMan“ wird, in Abhängigkeit des Source Mode, als blauer Balken dargestellt.	„PosSclMin“ und „PosSclMax“ skalieren den Bargraph.
Process value Bargraph	„PosFbk“ wird als blauer Balken dargestellt.	PosSclMin“ und „PosSclMax“ skalieren den Bargraph.
Position setpoint	„PosInt“ oder „PosMan“ wird, in Abhängigkeit des Source Mode, dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Process value	„PosFbk“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „PosUnit“ wird beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles) und Pipes (blaue Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 65 AnaVlv Symbolbeschreibung

12.1.8.2 hAnaVlvValue

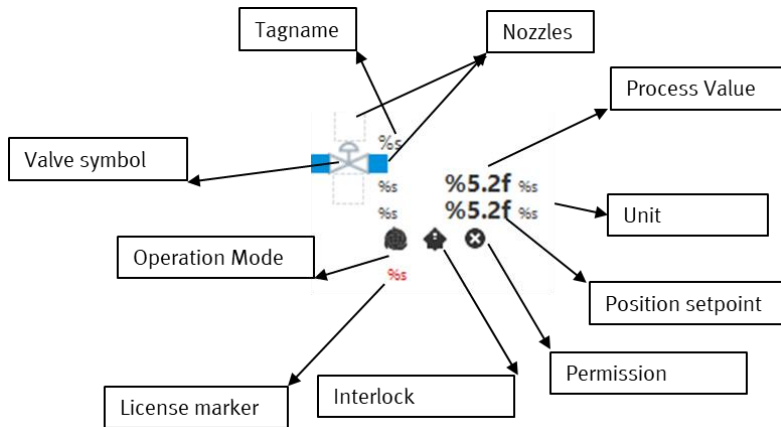


Fig. 69 Symbol hAnaVlvValue

12.1.8.3 vAnaVlvValue

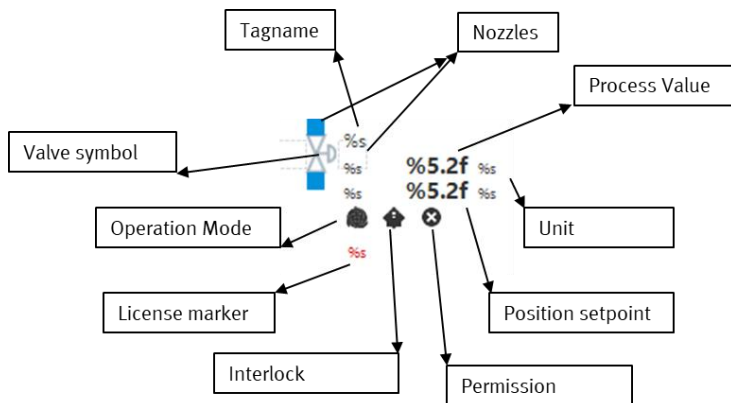


Fig. 70 Symbol vAnaVlvValue

12.1.8.4 hAnaVlvBargraph

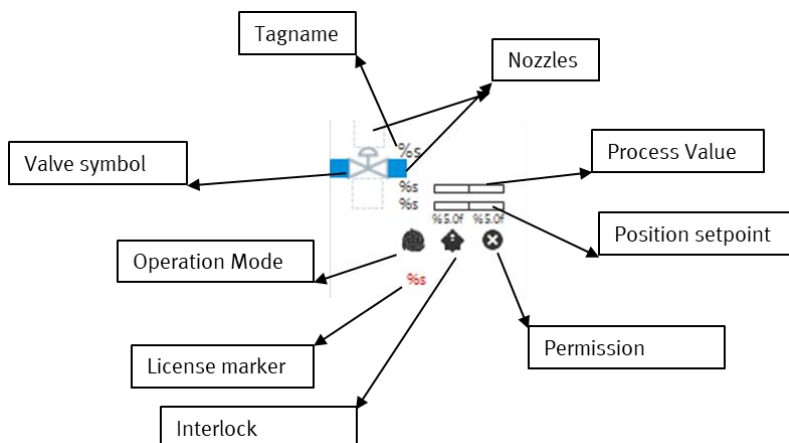


Fig. 71 Symbol hAnaVlvBargraph

12.1.8.5 vAnVlvBargraph

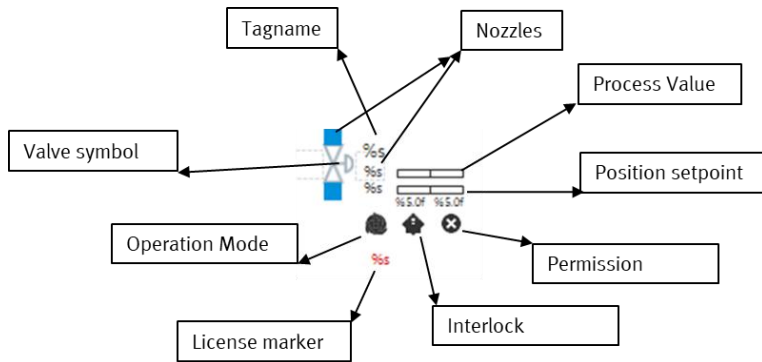


Fig. 72 Symbol vAnaVlvBargraph

12.2 Funktionsbaustein MonAnaVlv

12.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) MonAnaVlv erweitert den FB AnaVlv um das Feedback-Monitoring von dynamischen und statischen Fehlern sowie um eine Positionsüberwachung.

12.2.2 Erweiterte Funktionen

12.2.2.1 Feedback Monitoring aktivieren

Das Feedback-Monitoring (statisch, dynamisch und die Positionsüberwachung) kann über den Eingang am FB „MonEn“ (=TRUE) aktiviert werden. Ist „MonEn“ auf FALSE gesetzt oder wird im laufenden Betrieb auf FALSE gesetzt so werden alle Fehlerausgänge zurückgesetzt. Auch in der Visualisierung verschwindet das Alarm-Symbol und der Fehler muss auch nicht quittiert werden.

12.2.2.2 Eingang „MonSafePos“

Über den Eingang am FB „MonSafePos“ kann eingestellt werden, ob bei einem Fehler das Ventil die Sicherheitsposition anfährt (MonSafePos = TRUE) oder, ob der aktuelle Zustand gehalten werden soll (MonSafePos = FALSE). Bei beiden Varianten wird das Ausgangsbit SafePosAct auf TRUE gesetzt, damit der Ausnahmezustand erkannt wird. Als Standardeinstellung wird bei einem Fehler die Sicherheitsposition angefahren.

Soll nach einem Fehler der letzte Zustand gehalten werden oder befindet sich das Ventil schon im Sicherheitszustand, wird ein Fehler erst beseitigt, wenn die entsprechenden Feedbacksignale die richtigen Werte liefern.

Wird nach einem Fehler das Ventil in den Sicherheitszustand gefahren, wird der Fehler erst behoben, wenn nochmal in den Zustand, wo der Fehler aufgetreten ist, verfahren wird und die Feedbacksignale die richtigen Werte liefern.

Beispiel:

Das Ventil ist im Zustand Close und soll geöffnet werden. Nach dem Öffnen bleibt „OpenFbk“ auf FALSE. So wird das Ventil wieder in seinen Sicherheitszustand (hier Close) verfahren. Damit der Fehler verschwindet, muss das Ventil wieder geöffnet werden und „OpenFbk“ muss TRUE werden. Erst dann werden die Fehlerausgänge („StatErrAct“ und „DynErrAct“) wieder auf FALSE gesetzt.

12.2.2.3 Statische Fehler

Ein statischer Fehler liegt vor, wenn sich das Feedback-Signal ändert, ohne dass eine Änderung der Ansteuerung erfolgt ist.

Folgende statische Fehler können auftreten:

- „OpenFbk“ und „CloseFbk“ sind beide TRUE
- Zustand ist Open, „OpenFbk“ wird FALSE
- Zustand ist Close, „CloseFbk“ wird FALSE

Damit eine Störung nicht als verlorenes bzw. falsches Signal gewertet wird, muss dieser Fehlerzustand eine bestimmte Zeit anliegen damit es als Fehler gewertet wird. Diese Zeit kann über den Eingang am FB „MonStatTi“ eingestellt werden. Erst wenn diese Zeit abgelaufen ist und der Fehlerzustand immer noch anliegt, wird der Fehler erkannt und der Ausgang „StatErrAct“ wird auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

Hinweis:

Soll bei einem Fehler in die Safeposition gefahren werden so würden sich im Automatic Mode OpenAut und Safeposition ständig gegenseitig aufheben.

In diesem Fall wird die Safeposition angefahren und der OpMode wird auf Operator gestellt, da wegen des Fehlers ein Eingriff notwendig ist.

Der FB kann dann über das Faceplate oder über die Eingänge am FB wieder umgestellt werden.

12.2.2.4 Dynamische Fehler

Ein dynamischer Fehler liegt vor, wenn sich das Feedback-Signal nicht ändert, obwohl eine Änderung der Ansteuerung erfolgt ist.

Folgende dynamische Fehler können auftreten:

- Zustand soll sich von Open nach Close ändern- „CloseFbk“ bleibt auf FALSE
- Zustand soll sich von Close nach Open ändern- „OpenFbk“ bleibt auf FALSE

Damit das Feedbacksignal nicht sofort erwartet wird, da ein Ventil Zeit zum Schließen und Öffnen benötigt, kann über den Eingang am FB „MonDynTi“ eine Zeit eingestellt werden. Wird in dieser Zeit nicht das richtige Feedbacksignal erkannt wird der Ausgang „DynErrAct“ auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

Hinweis:

Soll bei einem Fehler in die Safeposition gefahren werden so würden sich im Automatic Mode OpenAut und Safeposition ständig gegenseitig aufheben.

In diesem Fall wird die Safeposition angefahren und der OpMode wird auf Operator gestellt, da wegen des Fehlers ein Eingriff notwendig ist.

Der FB kann dann über das Faceplate oder über die Eingänge am FB wieder umgestellt werden. Falls der Fehler nicht behoben wurde, so wird wieder in den Operator Mode gewechselt.

12.2.2.5 Positionsüberwachung

Wird eine Position über die Steuerbefehle Open und Close angefahren, so muss das Ventil die Position in einer bestimmten Zeit erreichen. Diese Zeit kann über den Eingang am FB „MonPosTi“ eingestellt werden. Zusätzlich kann ein Toleranzwert über den Eingang am FB „PosTolerance“ angegeben werden. Dieser Wert wird zur Sollposition addiert und subtrahiert, so dass sich ein Toleranzbereich errechnen lässt, in dem die Position als erreicht gilt. Erreicht das Ventil in der angegebenen Zeit nicht diesen Toleranzbereich so wird der Ausgang „MonPosErr“ auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

Hinweis:

Bei einem Positionsfehler wird nicht in die Safeposition gefahren, es werden nur die entsprechenden Fehlervariablen gesetzt.

12.2.2.6 Alarm Priorität

Folgende Priorität wird bei der Alarmanzeige in der Visualisierung eingehalten:

- Failed to open (höchste Priorität)
- Failed to close
- Both Feedbacks active
- Open Feedback lost
- Close Feedback lost
- Failed to reached position
- No alarm

12.2.2.7 Zeit „MonStatTi“, „MonDynTi“ und „MonPosTi“

Die Variablen „MonStatTi“, „MonDynTi“ und „MonPosTi“ werden in Sekunden angegeben und sind standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

12.2.2.8 Alarm quittieren

Liegt kein Fehler mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („MonAckOp“) oder über den Eingang „MonAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 66 Zustände Alarmsymbol

Hinweis:

- Der Eingang „MonAckAut“ ist flankengesteuert und reagiert nur auf ansteigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarmer gleichzeitig quittiert.
- Es können auch mehrere Fehler gleichzeitig anliegen – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

12.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				Pos < PosMin ⇒ Pos = PosMin
				Pos > PosMax ⇒ Pos = PosMax
				PosMin > PosMax
				PosSclMax < PosSclMin ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosSclMax
				PosSclMin < PosMin oder PosSclMin > PosMax ⇒ PosSclMin = PosMin
				PosSclMax < PosMin oder PosSclMax > PosMax ⇒ PosSclMax = PosMax
				PosSclMax – PosSclMin = 0 ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 67 MonAnaVlv spezifischer Worst Quality Code

12.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePosEn	BOOL	TRUE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePos	BOOL	FALSE	Safe position (1: PosMax 0: PosMin)
OSLevel	BYTE	0	OS Level Variable
MonEn	BOOL	FALSE	Enables dynamic, static and position Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s], only positive values
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s], only positive values
PosTolerance	REAL	0	Position Tolerance Parameter for calculation [PosUnit]
MonPosTi	REAL	0.5	Monitor Time until position is reached [s], only positive values

Tab. 68 MonAnaVlv persistente Variablen

12.2.5 Symbole

12.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Valve symbol	Grau: „CloseFbk“ ist aktiv Blau: „OpenFbk“ ist aktiv	
Alarm symbol	 : Alarm ist aktiv (höchste Priorität)	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Position setpoint Bargraph	„PosInt“ oder „PosMan“ wird, in Abhängigkeit des Source Mode, als blauer Balken dargestellt.	„PosSclMin“ und „PosSclMax“ skalieren den Bargraph.
Process value Bargraph	„PosFbk“ wird als blauer Balken dargestellt. den Bargraph.	„PosSclMin“ und „PosSclMax“ skalieren den Bargraph.
Position setpoint	„PosInt“ oder „PosMan“ wird, in Abhängigkeit des Source Mode, dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Process value	„PosFbk“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „PosUnit“ wird beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Unit	Einheit „PosUnit“ wird beim Sollwert und Prozesswert dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles) und Pipes (blaue Nozzles).	
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 69 MonAnaVlv Symbolbeschreibung

12.2.5.2 hMonAnaVlvValue

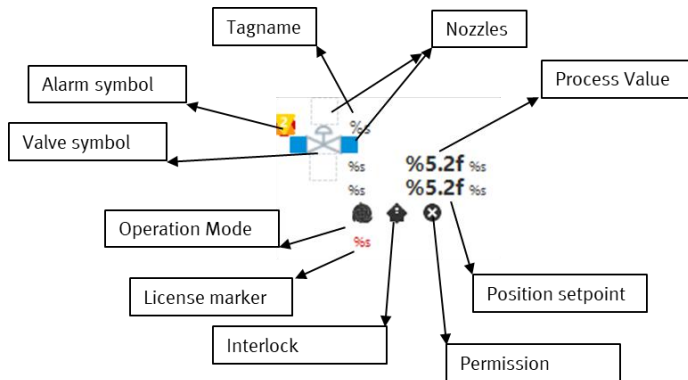


Fig. 73 Symbol hMonAnaVlvValue

12.2.5.3 vMonAnaVlvValue

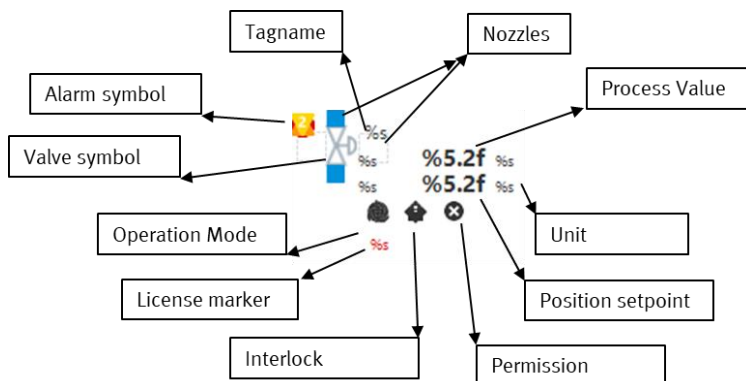


Fig. 74 Symbol vMonAnaVlvValue

12.2.5.4 hMonAnaVlvBargraph

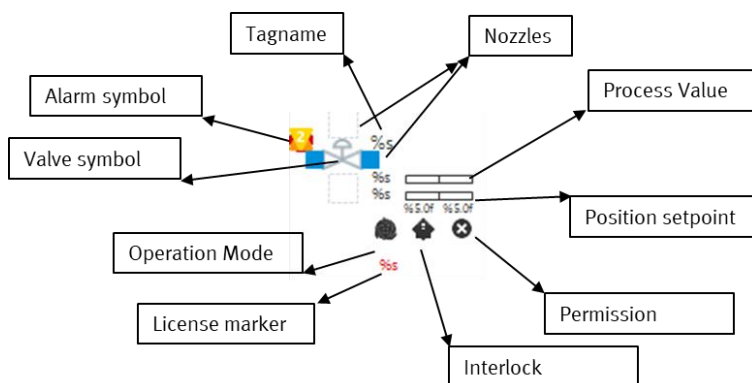


Fig. 75 Symbol hMonAnaVlvBargraph

12.2.5.5 vMonAnaVlvBargraph

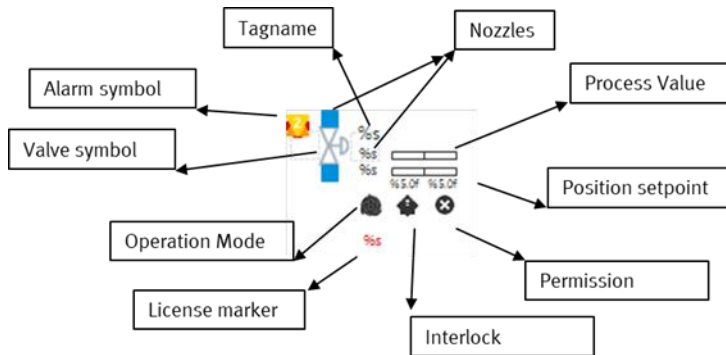


Fig. 76 Symbol vMonAnaVlvBargraph

12.2.6 Faceplates

12.2.6.1 Main

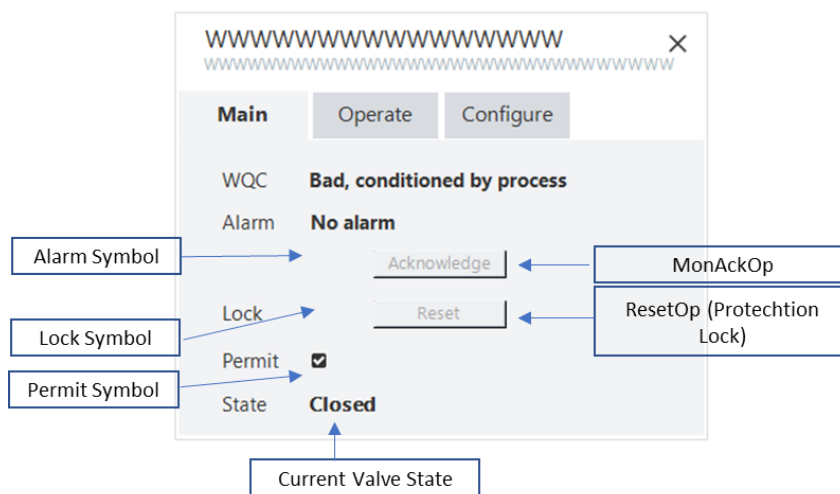


Fig. 77 MonAnaVlv Main Faceplate

12.2.6.2 Operate

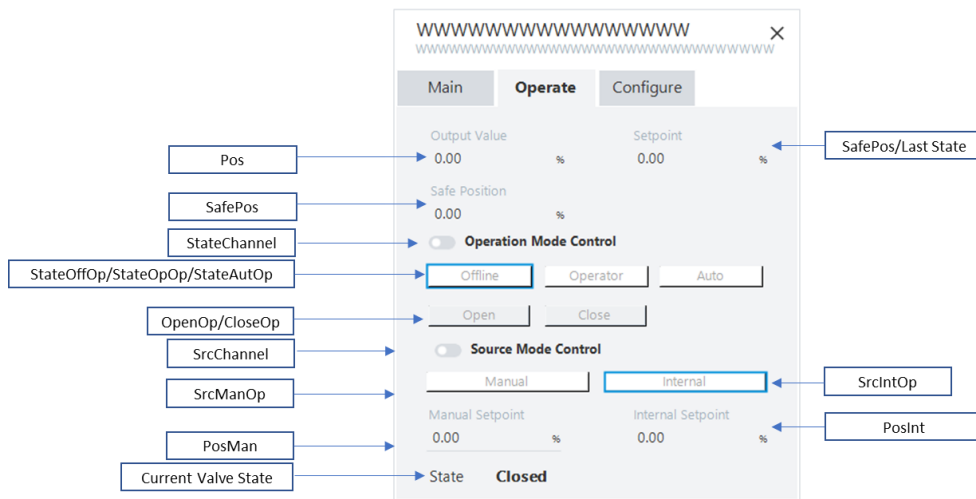


Fig. 78 MonAnaVlv Operate Faceplate

12.2.6.3 Configure

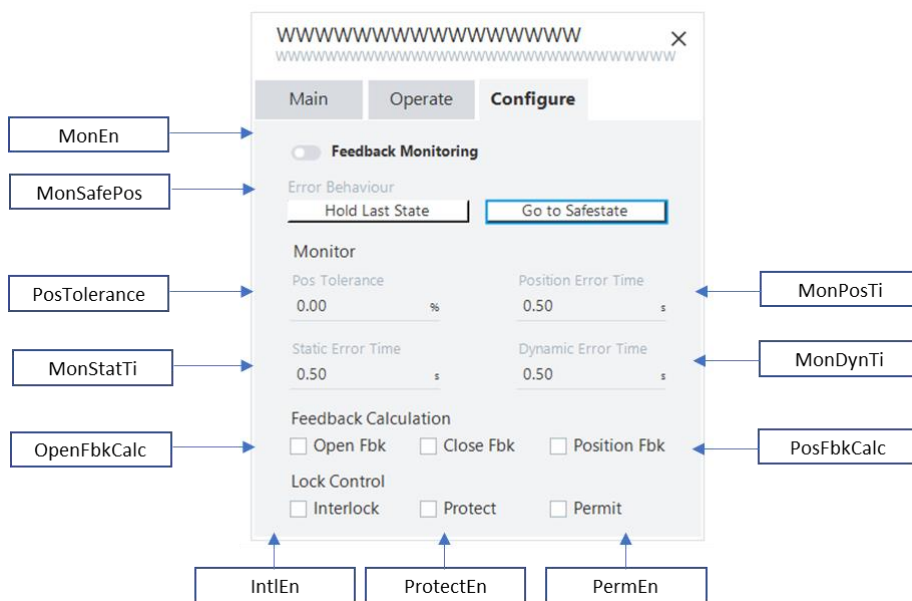


Fig. 79 MonAnaVlv Configure Faceplate

13 Bistabile Motoren

13.1 Funktionsbaustein BinDrv

13.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinDrv bietet die Möglichkeit einen Motor ein-bzw. auszuschalten sowie die Drehrichtung vorzugeben. Steuerungskommandos können entweder direkt am FB (Aut-Variablen) oder über ein Faceplate/OPC UA (Op-Variablen) gesetzt werden. Zusätzlich stehen Feedbacksignale zur Verfügung, welchen den tatsächlichen Antriebszustand anzeigen.

13.1.2 Grundfunktionen

13.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

13.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 70 Worst Quality Code Enumeration

13.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

13.1.3 Funktionen

13.1.3.1 Motorsteuerung

Der Motor kann entweder vorwärts, rückwärts betrieben oder gestoppt werden. Zur Steuerung des Motors stehen Eingänge am FB („FwdAut“, „RevAut“ und „StopAut“) und Variablen über OPC UA/Faceplate („FwdOp“, „RevOp“ und „StopOp“) bereit. Der Vorwärts- und Rückwärtslauf müssen jeweils über einen Eingang am FB „FwdEn“ und „RevEn“ (=TRUE) aktiviert werden. Das Steuerkommando wird im FB verarbeitet und an den entsprechenden Ausgang „FwdCtrl“ und „RevCtrl“ ausgegeben.

Ein Steuerbefehl wird abgegeben indem die oben genannten Variablen je nach Befehl auf TRUE gesetzt werden. Die Eingangsvariablen „FwdAut“, „RevAut“ und „StopAut“ sind flankengesteuert und reagieren auf ansteigende Flanken. „FwdOp“, „RevOp“ und „StopOp“ werden nach der Registrierung vom FB wieder auf FALSE gesetzt.

In der folgenden Tabelle ist aufgeführt wie sich die Ausgänge in Abhängigkeit der Sicherheitsposition und den Steuerkommandos verhalten.

Steuerkommando	Ausgänge
Stopp	FwdCtrl = 0 RevCtrl = 0
Fwd	FwdCtrl = 1 RevCtrl = 0
Rev	FwdCtrl = 0 RevCtrl = 1
Fwd, Rev, Stopp gleichzeitig	FwdCtrl = 0 RevCtrl = 0
Fwd, Rev gleichzeitig	Hold last state

Tab. 71 Steuerkommandos Motor

Der FB kann in drei Zustände Offline, Operator und Automatic betrieben werden. Die Zustände werden über den Operation Mode gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Folgende Steuerbefehle und Ausgänge werden im jeweiligen Zustand verarbeitet und gesetzt:

- Offline Mode: FwdCtrl = 0, RevCtrl = 0
- Operator Mode: „StopOp“, „FwdOp“ und „RevOp“ steuern „FwdCtrl“ und „RevCtrl“ (über Faceplate änderbar)
- Automatic Mode: „StopAut“, „FwdAut“ und „RevAut“ steuern „FwdCtrl“ und „RevCtrl“ (Eingänge am FB)

13.1.3.2 Sicherheitsposition „SafePos“

Über den Eingang „SafePos“ kann eingestellt werden, ob der Sicherheitszustand Running (SafePos = TRUE) oder Stopped (SafePos = FALSE) ist.

Damit keine Fehler beim Sicherheitszustand entsteht, müssen folgende Punkte beachtet werden:

- Wenn der Sicherheitszustand Running ist, muss unbedingt Vorwärts- oder Rückwärtslauf („FwdEn“ oder „RevEn“) aktiviert sein
- Wenn der Sicherheitszustand Running ist und der aktuelle Zustand ebenfalls Running ist dann wird im Fehlerfall dieser Zustand fortgeführt
- Wenn der Sicherheitszustand Running ist und der aktuelle Zustand Stopp ist so wird, falls der Vorwärtslauf aktiviert ist, dieser ausgeführt. Falls dieser nicht aktiviert ist, wird geprüft, ob der Rückwärtslauf aktiviert ist. Falls dieser auch nicht aktiviert ist, so bleibt der Zustand Stopp erhalten.

Die verschiedenen Zustände sind in folgender Tabelle zusammengefasst:

Sicherheitsposition	Aktueller Zustand	Fehlerzustand
SafePos = TRUE (Running)	FwdCtrl = 1 RevCtrl = 0	FwdCtrl = 1 RevCtrl = 0
SafePos = TRUE (Running)	FwdCtrl = 0 RevCtrl = 1	FwdCtrl = 0 RevCtrl = 1
SafePos = TRUE (Running)	FwdCtrl = 0 RevCtrl = 0	FwdEn = 1 -> FwdCtrl = 1 RevEn = 1 -> RevCtrl = 1 FwdEn = 0 und RevEn = 0 -> FwdCtrl = 0, RevCtrl = 0
SafePos = FALSE (Stopped)	FwdCtrl = 1 RevCtrl = 0	FwdCtrl = 0 RevCtrl = 0
SafePos = FALSE (Stopped)	FwdCtrl = 0 RevCtrl = 1	FwdCtrl = 0 RevCtrl = 0
SafePos = FALSE (Stopped)	FwdCtrl = 0 RevCtrl = 0	FwdCtrl = 0 RevCtrl = 0

Tab. 72 Sicherheitspositionen Motor

13.1.3.3 Feedbacksignale (simuliert, tatsächlicher Motorzustand)

Feedbacksignale werden automatisch vom FB selbst simuliert, wenn die Eingänge „FwdFbkCalc“ und/oder „RevFbkCalc“ auf TRUE gesetzt sind.

Falls keine Simulation gewünscht ist können die Eingänge „FwdFbkSignal“ und/oder „RevFbkSignal“ mit den tatsächlichen Feedbacksignalen des Motors verbunden werden. Zusätzlich müssen „FwdFbkCalc“ und „RevFbkCalc“ auf FALSE gesetzt werden. Wird eine Endlage erreicht, so wird der entsprechende Ausgang („FwdFbk“ bzw. „RevFbk“) auf

TRUE gesetzt. Sind beide Feedbacks auf FALSE gesetzt so wurde der Stopp-Zustand erreicht. Als Standardeinstellung werden die richtigen Feedbacksignale über „FwdFbkSignal“ und/oder „RevFbkSignal“ verarbeitet.

13.1.3.4 Trip

Das Trip-Signal kann über den Eingang am FB „TripFbk“ an den FB weitergegeben werden. Dieses Signal wird zur weiteren Verarbeitung an den Ausgang „Trip“ weitergeleitet.

13.1.3.5 Lock-Funktionen

13.1.3.5.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

Hinweis:

Ist die Safeposition auf „laufen“ gesetzt so kann auch im Permit zwischen Vorwärts- und Rückwärtslauf gewechselt werden. Nur gestoppt kann der Motor nicht werden. Falls das Verhalten nicht gewünscht ist, so kann über „FwdEn“ und „RevEn“ jeweils eine Richtung deaktiviert werden.

13.1.3.5.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

13.1.3.5.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate (ResetOp) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf ansteigende Flanken.

13.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 73 BinDrv spezifischer Worst Quality Code

13.1.5 Operation Mode

13.1.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

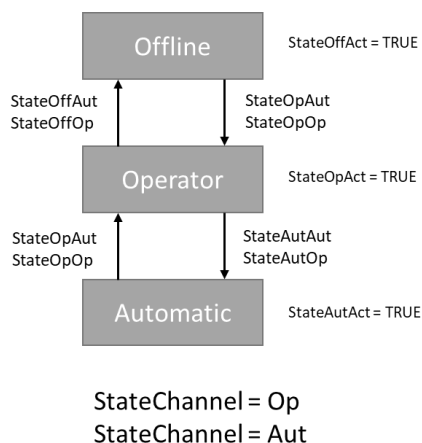


Fig. 80 Zustandsdiagramm Operation Mode

13.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

13.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

13.1.6 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable

Tab. 74 BinDrv persistente Variablen

13.1.7 Symbole

13.1.7.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Drive symbol	Grau: Motor ist im Stopp Blau: Motor läuft	
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Drehrichtung	 : Motor läuft vorwärts  : Motor läuft rückwärts	Bei Stopp wird kein Symbol angezeigt.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles), Measurement Line (Nozzle mit durchgehender Linie) und Pipes (blaue Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 75 BinDrv Symbolbeschreibung

13.1.7.2 dBinDrvPumpValue/uBinDrvPumpValue/rBinDrvPumpValue/lBinDrvPumpValue

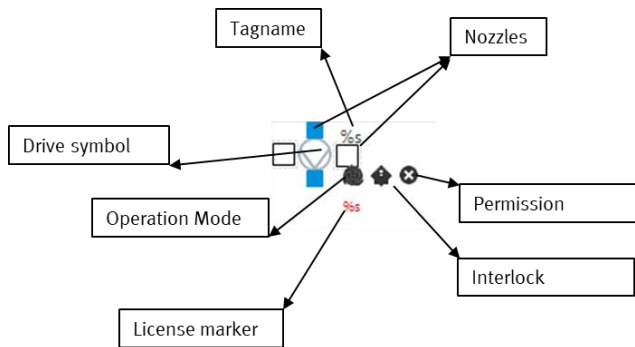


Fig. 81 Symbols BinDrv Pump

13.1.7.3 hBinDrvValue

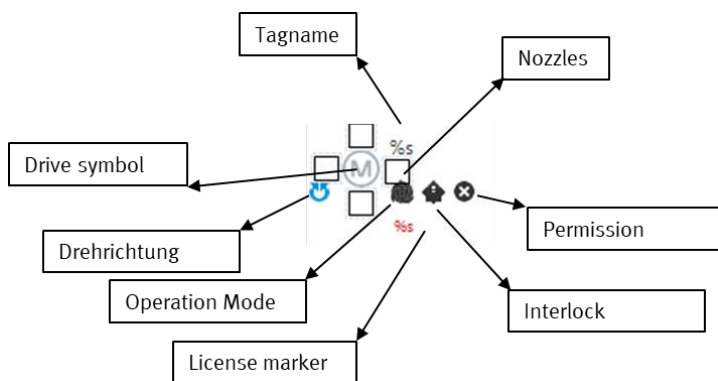


Fig. 82 Symbol hBinDrvValue

13.2 Funktionsbaustein MonBinDrv

13.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) MonBinDrv erweitert den FB BinDrv um das Feedback-Monitoring von dynamischen und statischen Fehlern.

13.2.2 Erweiterte Funktionen

13.2.2.1 Feedback Monitoring aktivieren

Das Feedback-Monitoring (statisch und dynamisch) kann über den Eingang am FB „MonEn“ (=TRUE) aktiviert werden. Ist „MonEn“ auf FALSE gesetzt oder wird im laufenden Betrieb auf FALSE gesetzt so werden alle Fehlerausgänge zurückgesetzt. Auch in der Visualisierung verschwindet, das Alarm-Symbol und der Fehler muss nicht quittiert werden.

13.2.2.2 Eingang „MonSafePos“

Über den Eingang am FB „MonSafePos“ kann eingestellt werden, ob bei einem Fehler der Motor den Sicherheitszustand einnimmt (MonSafePos = TRUE) oder, ob der aktuelle Zustand gehalten werden soll (MonSafePos = FALSE). Bei beiden Varianten wird das Ausgangsbit SafePosAct auf TRUE gesetzt, damit der Ausnahmezustand erkannt wird. Als Standardeinstellung wird bei einem Fehler die Sicherheitsposition angefahren.

Soll nach einem Fehler der letzte Zustand gehalten werden oder befindet sich der Motor schon im Sicherheitszustand, wird ein Fehler erst beseitigt, wenn die entsprechenden Feedbacksignale die richtigen Werte liefern.

Wird nach einem Fehler der Motor in den Sicherheitszustand gefahren, wird der Fehler erst behoben, wenn nochmal in den Zustand, wo der Fehler aufgetreten ist, verfahren wird und die Feedbacksignale die richtigen Werte liefern.

Beispiel:

Der Motor ist im Stopp und soll gestartet werden. Nach dem Start bleibt „FwdFbk“ auf FALSE. So wird der Motor wieder in seinen Sicherheitszustand (hier Stopp) versetzt. Damit der Fehler verschwindet, muss der Motor wieder gestartet werden und „FwdFbk“ muss TRUE werden. Erst dann werden die Fehlerausgänge (StatErrAct und DynErrAct) wieder auf FALSE gesetzt.

13.2.2.3 Statische Fehler

Ein statischer Fehler liegt vor, wenn sich das Feedback-Signal ändert, ohne dass eine Änderung der Ansteuerung erfolgt ist.

Folgende statische Fehler können auftreten:

- „FwdFbk“ und „RevFbk“ sind beide TRUE
- Zustand ist Run, „FwdFbk“/ „RevFbk“ wird FALSE
- Zustand ist Stopp, „FwdFbk“/ „RevFbk“ wird TRUE

Damit eine Störung nicht als verlorenes bzw. falsches Signal gewertet wird, muss dieser Fehlerzustand eine bestimmte Zeit anliegen damit es als Fehler gewertet wird. Diese Zeit kann über den Eingang am FB „MonStatTi“ eingestellt werden. Erst wenn diese Zeit abgelaufen ist und der Fehlerzustand immer noch anliegt, wird der Fehler erkannt und der Ausgang „StatErrAct“ wird auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

13.2.2.4 Dynamische Fehler

Ein dynamischer Fehler liegt vor, wenn sich das Feedbacksignal nicht ändert, obwohl eine Änderung der Ansteuerung erfolgt ist.

Folgende dynamische Fehler können auftreten:

- Zustand soll sich von Run nach Stopp ändern- „FwdFbk“/ „RevFbk“ bleibt auf TRUE
- Zustand soll sich von Stopp in Run ändern- „FwdFbk“/ „RevFbk“ bleibt FALSE
- Zustand läuft vorwärts soll sich in Zustand lauf rückwärts ändern- „RevFbk“ bleibt FALSE

Damit das Feedbacksignal nicht sofort erwartet wird, da ein Motor Zeit zum Anlaufen oder Stoppen benötigt, kann über den Eingang am FB „MonDynTi“ eine Zeit eingestellt werden. Wird in dieser Zeit nicht das richtige Feedbacksignal erkannt wird der Ausgang „DynErrAct“ auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

13.2.2.5 Alarm Priorität

Folgende Priorität wird bei der Alarmanzeige in der Visualisierung eingehalten:

- Both Feedbacks active (höchste Priorität)
- Forward Feedback lost
- Reverse Feedback lost
- Stop Feedback lost
- Failed to run forward
- Failed to run reserve
- Failed to stop
- No alarm

13.2.2.6 Zeit „MonStatTi“ und „MonDynTi“

Die Variablen „MonStatTi“ und „MonDynTi“ werden in Sekunden angegeben und sind standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

13.2.2.7 Alarm quittieren

Liegt kein Fehler mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („MonAckOp“) oder über den Eingang „MonAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 76 Zustände Alarmsymbol

Hinweis:

- Der Eingang „MonAckAut“ ist flankengesteuert und reagiert nur auf ansteigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarme gleichzeitig quittiert.
- Es können auch mehrere Fehler gleichzeitig anliegen – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

13.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 77 MonBinDrv spezifischer Worst Quality Code

13.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable
MonEn	BOOL	FALSE	Enables dynamic, static and position Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s], only positive values
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s], only positive values

Tab. 78 MonBinDrv persistente Variablen

13.2.5 Symbole

13.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben	Max 16 Zeichen
Alarm symbols	 : Alarm ist aktiv	Siehe Beschreibung Funktionsbaustein und Alarmfunktion
Drive symbol	Grau: Motor ist im Stopp Blau: Motor läuft	
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion
Drehrichtung	 : Motor läuft vorwärts  : Motor läuft rückwärts	Bei Stopp wird kein Symbol angezeigt
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles), Measurement Line (Nozzle mit durchgehender Linie) und Pipes (blaue Nozzles)	Wichtig für den MTP Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“	Unsichtbar bis Lizenz abgelaufen

Tab. 79 MonBinDrv Symbolbeschreibung

13.2.5.2 dMonBinDrvPumpValue/uMonBinDrvPumpValue/rMonBinDrvPumpValue/lMonBinDrvPumpValue

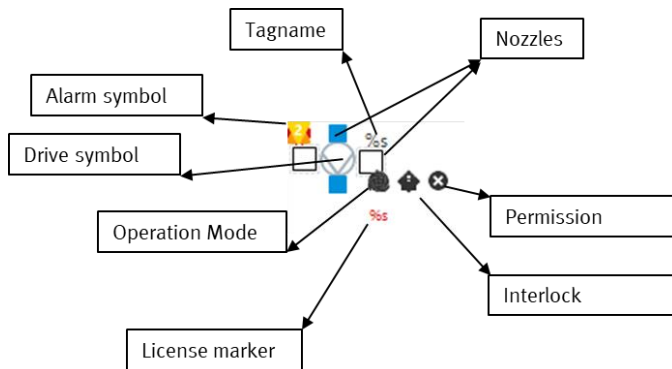


Fig. 83 Symbole MonBinDrv Pump

13.2.5.3 hMonBinDrvValue

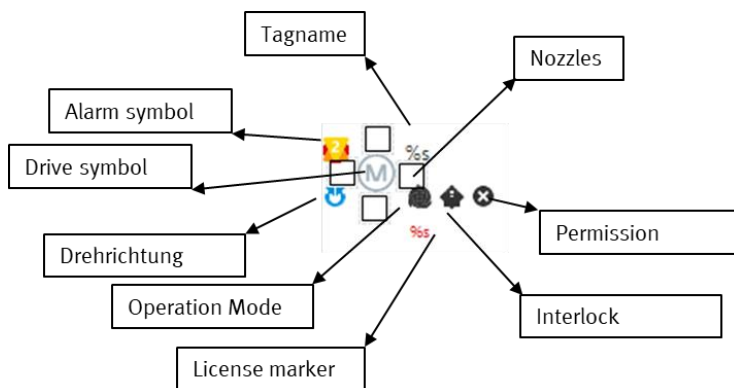


Fig. 84 Symbol hMonBinDrvValue

13.2.6 Faceplates

13.2.6.1 Main

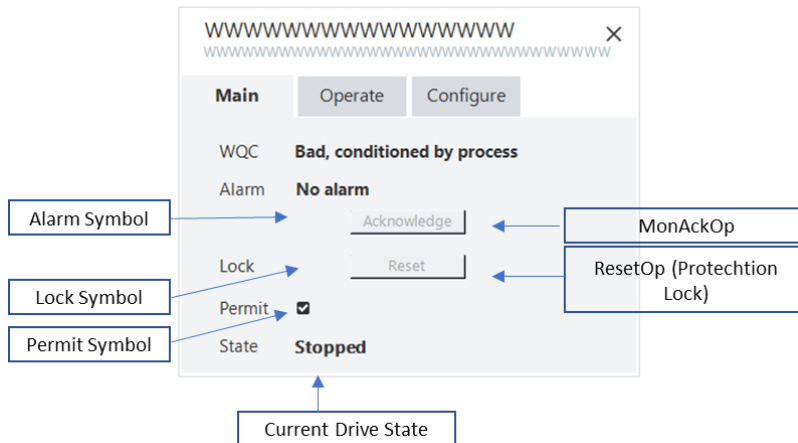


Fig. 85 MonBinDrv Main Faceplate

13.2.6.2 Operate

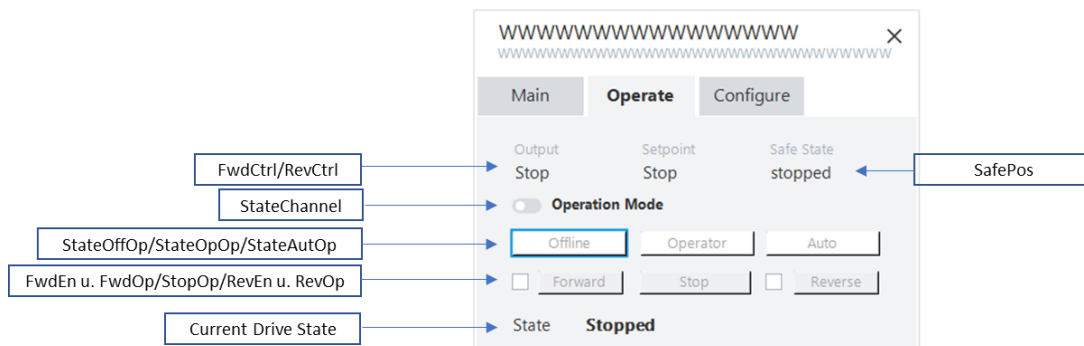


Fig. 86 MonBinDrv Operate Faceplate

13.2.6.3 Configure

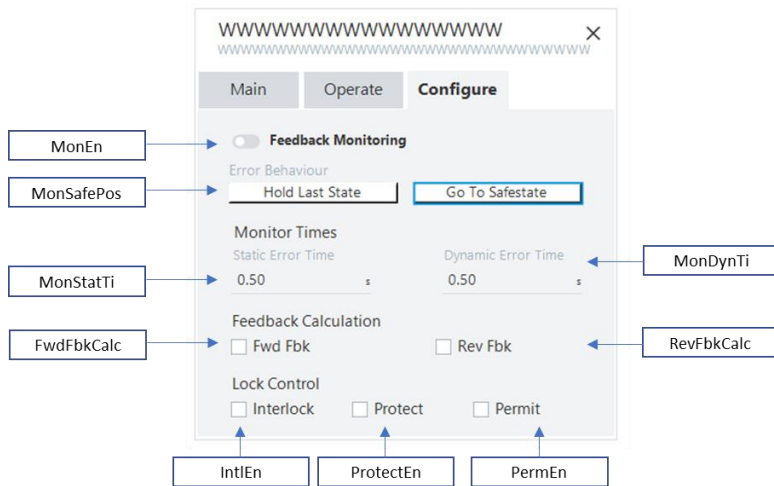


Fig. 87 MonBinDrv Configure Faceplate

14 Regelbare Motoren

14.1 Funktionsbaustein AnaDrv

14.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaDrv bietet die Möglichkeit einen Motor ein-bzw. auszuschalten sowie die Drehrichtung und eine variable Geschwindigkeit vorzugeben. Steuerungskommandos können entweder direkt am FB (Aut-Variablen) oder über ein Faceplate/OPC UA (Op-Variablen) gesetzt werden. Zusätzlich stehen Feedbacksignale zur Verfügung, welchen den tatsächlichen Antriebszustand anzeigen.

14.1.2 Grundfunktionen

14.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

14.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 80 Worst Quality Code Enumeration

14.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

14.1.3 Funktionen

14.1.3.1 Sollwertvorgabe Geschwindigkeit „RpmMan“ und „RpmInt“

Wird der Motor gestartet, so kann zusätzlich eine Geschwindigkeit eingestellt werden. Der Sollwert der Geschwindigkeit kann über zwei Quellen angegeben werden. Über den Eingang „RpmInt“ kann direkt am FB der Sollwert vorgegeben werden. Soll der Sollwert über OPC UA oder das Faceplate eingestellt werden, so wird das Eingabefeld „Manual Setpoint“ genutzt. Welcher der beiden Eingänge gerade für die Geschwindigkeit herangezogen wird steuert der Source Mode. Bei der Initialisierung ist die interne Quelle und somit „RpmInt“ aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erklärt.

Der eingestellte Sollwert wird unter Berücksichtigung der Grenzwerte an den Ausgang „Rpm“ ausgegeben.

14.1.3.2 Begrenzung Sollwert „RpmInt“ und „RpmMan“

Über die Eingänge am FB „RpmMin“ und „RpmMax“ können Grenzwerte für die Geschwindigkeit eingegeben werden. Ist der angegebene Sollwert („RpmInt“ bzw. „RpmMan“) größer „RpmMax“ oder kleiner „RpmMin“ so wird der Sollwert auf den entsprechenden Grenzwert begrenzt. Zusätzlich muss darauf geachtet werden, dass „RpmMin“ nicht größer ist als „RpmMax“. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

14.1.3.3 Einheit „RpmInt“, „RpmMan“ und „Rpm“

Für die Position kann über den Eingang am FB „RpmUnit“ eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

14.1.3.4 Motorsteuerung

Der Motor kann entweder vorwärts, rückwärts betrieben oder gestoppt werden. Zur Steuerung des Motors stehen Eingänge am FB („FwdAut“, „RevAut“ und „StopAut“) und Variablen über OPC UA/Faceplate („FwdOp“, „RevOp“ und „StopOp“) bereit. Der Vorwärts- und Rückwärtslauf muss jeweils über einen Eingang am FB „FwdEn“ und „RevEn“ (=TRUE) aktiviert werden. Das Steuerkommando wird im FB verarbeitet und an den entsprechenden Ausgang „FwdCtrl“ und „RevCtrl“ ausgegeben.

Ein Steuerbefehl wird abgegeben indem die oben genannten Variablen je nach Befehl auf TRUE gesetzt werden. Die Eingangsvariablen „FwdAut“, „RevAut“ und „StopAut“ sind flankengesteuert und reagieren auf ansteigende Flanken. „FwdOp“, „RevOp“ und „StopOp“ werden nach der Registrierung vom FB wieder auf FALSE gesetzt.

In der folgenden Tabelle ist aufgeführt wie sich die Ausgänge in Abhängigkeit der Sicherheitsposition und den Steuerkommandos verhalten.

Steuerkommando	Ausgänge
Stopp	FwdCtrl = 0 RevCtrl = 0
Fwd	FwdCtrl = 1 RevCtrl = 0
Rev	FwdCtrl = 0 RevCtrl = 1
Fwd, Rev, Stopp gleichzeitig	FwdCtrl = 0 RevCtrl = 0
Fwd, Rev gleichzeitig	Hold last state

Tab. 81 Steuerkommandos Motor

Der FB kann in drei Zustände Offline, Operator und Automatic betrieben werden. Die Zustände werden über den Operation Mode gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Folgende Steuerbefehle und Ausgänge werden im jeweiligen Zustand verarbeitet und gesetzt:

- Offline Mode: FwdCtrl = 0, RevCtrl = 0
- Operator Mode: „StopOp“, „FwdOp“ und „RevOp“ steuern „FwdCtrl“ und „RevCtrl“ (über Faceplate änderbar)
- Automatic Mode: „StopAut“, „FwdAut“ und „RevAut“ steuern „FwdCtrl“ und „RevCtrl“ (Eingänge am FB)

14.1.3.5 Sicherheitsposition „SafePos“

Über den Eingang „SafePos“ kann eingestellt werden, ob der Sicherheitszustand Running (SafePos = TRUE) oder Stopped (SafePos = FALSE) ist.

Damit keine Fehler beim Sicherheitszustand entsteht, müssen folgende Punkte beachtet werden:

- Wenn der Sicherheitszustand Running ist, muss unbedingt Vorwärts- oder Rückwärtslauf („FwdEn“ oder „RevEn“) aktiviert sein
- Wenn der Sicherheitszustand Running ist und der aktuelle Zustand ebenfalls Running ist dann wird im Fehlerfall dieser Zustand fortgeführt
- Wenn der Sicherheitszustand Running ist und der aktuelle Zustand Stopp ist so wird, falls der Vorwärtslauf aktiviert ist, dieser ausgeführt. Falls dieser nicht aktiviert ist, wird geprüft, ob der Rückwärtslauf aktiviert ist. Falls dieser auch nicht aktiviert ist, so bleibt der Zustand Stopp erhalten.

Die verschiedenen Zustände sind in folgender Tabelle zusammengefasst:

Sicherheitsposition	Aktueller Zustand	Fehlerzustand
SafePos = TRUE (Running)	FwdCtrl = TRUE RevCtrl = FALSE	FwdCtrl = TRUE RevCtrl = FALSE
SafePos = TRUE (Running)	FwdCtrl = FALSE RevCtrl = TRUE	FwdCtrl = FALSE RevCtrl = TRUE
SafePos = TRUE (Running)	FwdCtrl = FALSE RevCtrl = FALSE	FwdEn = TRUE -> FwdCtrl = TRUE RevEn = TRUE -> RevCtrl = TRUE FwdEn = FALSE und RevEn = FALSE -> FwdCtrl = FALSE, RevCtrl = FALSE
SafePos = FALSE (Stopped)	FwdCtrl = TRUE RevCtrl = FALSE	FwdCtrl = FALSE RevCtrl = FALSE
SafePos = FALSE (Stopped)	FwdCtrl = FALSE RevCtrl = TRUE	FwdCtrl = FALSE RevCtrl = FALSE
SafePos = FALSE (Stopped)	FwdCtrl = FALSE RevCtrl = FALSE	FwdCtrl = FALSE RevCtrl = FALSE

Tab. 82 Sicherheitspositionen Motor

14.1.3.6 Feedbacksignale (simuliert, tatsächlicher Motorzustand)

Feedbacksignale werden automatisch vom FB selbst simuliert, wenn die Eingänge „FwdFbkCalc“ und/oder „RevFbkCalc“ auf TRUE gesetzt sind.

Falls keine Simulation gewünscht ist, können die Eingänge „FwdFbkSignal“ und/oder „RevFbkSignal“ mit den tatsächlichen Feedbacksignalen des Motors verbunden werden. Zusätzlich müssen „FwdFbkCalc“ und „RevFbkCalc“ auf FALSE gesetzt werden. Wird eine Endlage erreicht, so wird der entsprechende Ausgang („FwdFbk“ bzw. „RevFbk“) auf TRUE gesetzt. Sind beide Feedbacks auf FALSE gesetzt so wurde der Stopp-Zustand erreicht. Als Standardeinstellung werden die richtigen Feedbacksignale über „FwdFbkSignal“ und/oder „RevFbkSignal“ verarbeitet.

Für die Geschwindigkeit kann das Feedbacksignal ebenfalls über einen Eingang am FB „RpmFbkCalc“ (=TRUE) simuliert werden. Soll das Feedbacksignal vom Motor verwendet werden, so kann das tatsächliche Signal über den Eingang am FB „RpmFbkSignal“ angebunden werden. Dazu muss der Eingang „RpmFbkCalc“ auf FALSE gesetzt werden.

14.1.3.7 Trip

Das Trip-Signal kann über den Eingang am FB „TripFbk“ an den FB weitergegeben werden. Dieses Signal wird zur weiteren Verarbeitung an den Ausgang „Trip“ weitergeleitet.

14.1.3.8 Lock-Funktionen

14.1.3.8.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition/Sicherheitswert angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

Hinweis:

Ist die Safeposition auf „laufen“ gesetzt so kann auch im Permit zwischen Vorwärts- und Rückwärtslauf gewechselt werden. Nur gestoppt kann der Motor nicht werden. Falls das Verhalten nicht gewünscht ist, so kann über „FwdEn“ und „RevEn“ jeweils eine Richtung deaktiviert werden. Die Drehzahl kann ebenfalls noch variiert werden.

14.1.3.8.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

14.1.3.8.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition/Sicherheitswert hat wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate (ResetOp) erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist. Befindet sich der FB im Sicherheitszustand so wird „SafePosAct“ auf TRUE gesetzt.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Automatic) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf ansteigende Flanken.

14.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				$Rpm < RpmMin$ ⇒ $Rpm = RpmMin$
				$Rpm > RpmMax$ ⇒ $Rpm = RpmMax$
				$RpmMin > RpmMax$
				$RpmSclMax < RpmSclMin$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax < RpmMin$ oder $RpmSclMax > RpmMax$ ⇒ $RpmSclMax = RpmMax$
				$RpmSclMin < RpmMin$ oder $RpmSclMin > RpmSclMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax - RpmSclMin = 0$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 83 AnaDrv spezifischer Worst Quality Code

14.1.5 Source Mode

14.1.5.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang SrcIntAct auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit SrcManAct auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

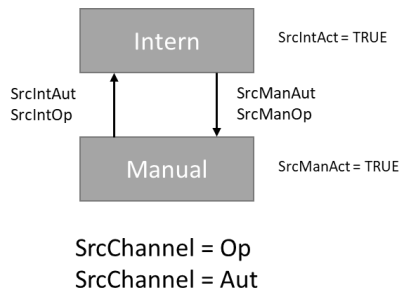


Fig. 88 Zustandsdiagramm Source Mode

14.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

14.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen SrcIntOp, SrcManOp, SrcIntAut oder SrcManAut an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

14.1.6 Operation Mode

14.1.6.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

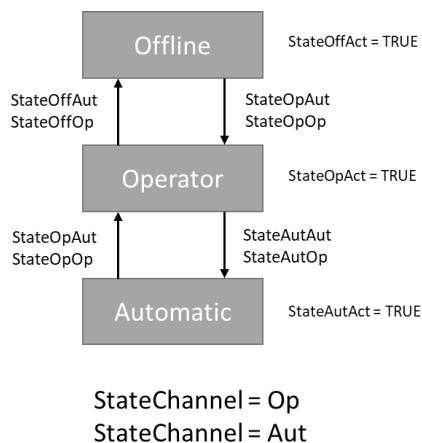


Fig. 89 Zustandsdiagramm Operation Mode

14.1.6.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

14.1.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

14.1.7 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable

Tab. 84 AnaDrv persistente Variablen

14.1.8 Symbole

14.1.8.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Drive symbol	Grau: Motor ist im Stopp Blau: Motor läuft	
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	 : kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Drehrichtung	 : Motor läuft vorwärts  : Motor läuft rückwärts	Bei Stopp wird kein Symbol angezeigt.
Rpm	Variable „RpmFbk“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	„RpmUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles), Measurement Line (Nozzle mit durchgehender Linie) und Pipes (blaue Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 85 AnaDrv Symbolbeschreibung

14.1.8.2 dAnaDrvPumpValue/uAnaDrvPumpValue/rAnaDrvPumpValue/lAnaDrvPumpValue

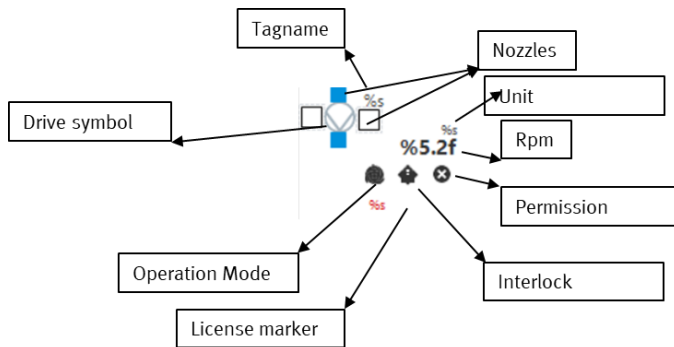


Fig. 90 Symbole AnaDrv Pump

14.1.8.3 hAnaDrvMotorValue

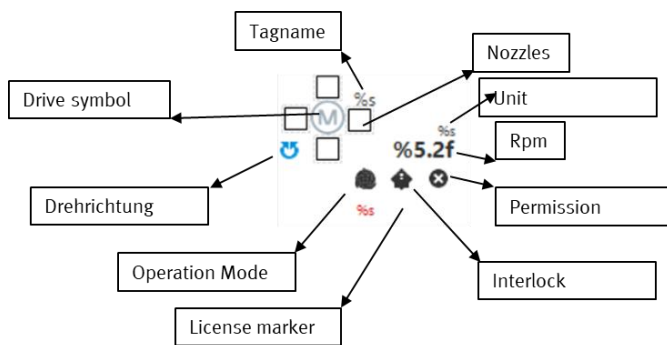


Fig. 91 Symbol hAnaDrvMotorValue

14.2 Funktionsbaustein MonAnaDrv

14.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) MonAnaDrv erweitert den FB AnaDrv um das Feedback-Monitoring von dynamischen und statischen Fehlern und einer Geschwindigkeitsüberwachung.

14.2.2 Erweiterte Funktionen

14.2.2.1 Feedback Monitoring aktivieren

Das Feedback-Monitoring (statisch und dynamisch) kann über den Eingang am FB „MonEn“ (=TRUE) aktiviert werden. Ist „MonEn“ auf FALSE gesetzt oder wird im laufenden Betrieb auf FALSE gesetzt so werden alle Fehlerausgänge zurückgesetzt. Auch in der Visualisierung verschwindet das Alarm-Symbol und der Fehler muss nicht quittiert werden.

14.2.2.2 Eingang „MonSafePos“

Über den Eingang am FB „MonSafePos“ kann eingestellt werden, ob bei einem Fehler der Motor den Sicherheitszustand einnimmt (MonSafePos = TRUE) oder, ob der aktuelle Zustand gehalten werden soll (MonSafePos = FALSE). Bei beiden Varianten wird das Ausgangsbit „SafePosAct“ auf TRUE gesetzt, damit der Ausnahmezustand erkannt wird. Als Standardeinstellung wird bei einem Fehler die Sicherheitsposition angefahren.

Soll nach einem Fehler der letzte Zustand gehalten werden oder befindet sich der Motor schon im Sicherheitszustand, wird ein Fehler erst beseitigt, wenn die entsprechenden Feedbacksignale die richtigen Werte liefern.

Wird nach einem Fehler der Motor in den Sicherheitszustand gefahren, wird der Fehler erst behoben, wenn nochmal in den Zustand wo der Fehler aufgetreten ist, verfahren wird und die Feedbacksignale die richtigen Werte liefern.

Beispiel:

Der Motor ist im Stopp und soll gestartet werden. Nach dem Start bleibt „FwdFbk“ auf FALSE. So wird der Motor wieder in seinen Sicherheitszustand (hier Stopp) versetzt. Damit der Fehler verschwindet, muss der Motor wieder gestartet werden und „FwdFbk“ muss TRUE werden. Erst dann werden die Fehlerausgänge („StatErrAct“ und „DynErrAct“) auf FALSE gesetzt.

14.2.2.3 Statische Fehler

Ein statischer Fehler liegt vor, wenn sich das Feedback-Signal ändert, ohne dass eine Änderung der Ansteuerung erfolgt ist.

Folgende statische Fehler können auftreten:

- „FwdFbk“ und „RevFbk“ sind beide TRUE
- Zustand ist Run, „FwdFbk“/ „RevFbk“ wird FALSE
- Zustand ist Stopp, „FwdFbk“/ „RevFbk“ wird TRUE

Damit eine Störung nicht als verlorenes bzw. falsches Signal gewertet wird, muss dieser Fehlerzustand eine bestimmte Zeit anliegen damit es als Fehler gewertet wird. Diese Zeit kann über den Eingang am FB „MonStatTi“ eingestellt werden. Erst wenn diese Zeit abgelaufen ist und der Fehlerzustand immer noch anliegt, wird der Fehler erkannt und der Ausgang „MonStatErr“ wird auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

14.2.2.4 Dynamische Fehler

Ein dynamischer Fehler liegt vor, wenn sich das Feedbacksignal nicht ändert, obwohl eine Änderung der Ansteuerung erfolgt ist.

Folgende dynamische Fehler können auftreten:

- Zustand soll sich von Run nach Stopp ändern- „FwdFbk“/ „RevFbk“ bleibt auf TRUE
- Zustand soll sich von Stopp in Run ändern- „FwdFbk“/ „RevFbk“ bleibt FALSE
- Zustand lauf vorwärts soll sich in Zustand lauf rückwärts ändern- „RevFbk“ bleibt FALSE

Damit das Feedbacksignal nicht sofort erwartet wird, da ein Motor Zeit zum Anlaufen oder Stoppen benötigt, kann über den Eingang am FB „MonDynTi“ eine Zeit eingestellt werden. Wird in dieser Zeit nicht das richtige Feedbacksignal erkannt wird der Ausgang „DynErrAct“ auf TRUE gesetzt.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

14.2.2.5 Geschwindigkeitsüberwachung

Mit dieser Funktion wird die Geschwindigkeit „RpmFbk“ überwacht. Es kann ein unterer Grenzwert „RpmALLim“ und ein oberer Grenzwert „RpmAHLim“ eingegeben werden, welche die Geschwindigkeit „RpmFbk“ nicht unterschreiten bzw. überschreiten darf.

Die Grenzwerte stehen als Eingänge am FB zur Verfügung und können separat über die Eingänge „RpmALEn“ und „RpmAHEn“ aktiviert werden. Zusätzlich kann über den Eingang am FB „MonRpmTi“ eine Zeit vorgegeben werden. Wird eine Grenzwertverletzung erkannt, so muss diese erst die Zeit von „MonRpmTi“ anliegen, damit sie als Fehler erkannt wird. Andernfalls wird es als flatterndes Signal angesehen und es wird kein Alarm ausgelöst.

Ein Alarm wird über das zugehörige Ausgangsbit ausgegeben („RpmALerr“ und „RpmAHerr“), es wird jedoch keine Sicherheitsposition angefahren. Über den Ausgang „RpmErr“ wird die Differenz zwischen „Rpm“ und „RpmFbk“ ausgegeben.

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

Werden Grenzwerte falsch gesetzt, (z.B. unterer Grenzwert > oberer Grenzwert) wird der WQC entsprechend gesetzt.

Hinweis:

- Wird ein Motor gestoppt, so wird „RpmFbk“ nicht auf die Grenzwerte überprüft, sondern ob der Wert < 0.5 ist.

14.2.2.6 Alarm Priorität

Folgende Priorität wird bei der Alarmanzeige in der Visualisierung eingehalten:

- Both Feedbacks active (höchste Priorität)
- Forward Feedback lost
- Reverse Feedback lost
- Stop Feedback lost
- Failed to run forward
- Failed to run reserve
- Failed to stop
- Rpm High alarm active
- RpmLow alarm active
- No alarm

14.2.2.7 Zeit „MonStatTi“, „MonDynTi“, „MonRpmTi“

Die Variablen „MonStatTi“, „MonDynTi“ und „MonRpmTi“ werden in Sekunden angegeben und sind standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

14.2.2.8 Alarm quittieren

Liegt kein Fehler mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („MonAckOp“) oder über den Eingang „MonAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 86 Zustände Alarmsymbol

Hinweis:

- Der Eingang „MonAckAut“ ist flankengesteuert und reagiert nur auf ansteigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarmer gleichzeitig quittiert.
- Es können auch mehrere Fehler gleichzeitig anliegen – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

14.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				$Rpm < RpmMin$ ⇒ $Rpm = RpmMin$
				$Rpm > RpmMax$ ⇒ $Rpm = RpmMax$
				$RpmMin > RpmMax$
				$RpmSclMax < RpmSclMin$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax < RpmMin$ oder $RpmSclMax > RpmMax$ ⇒ $RpmSclMax = RpmMax$
				$RpmSclMin < RpmMin$ oder $RpmSclMin > RpmSclMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax - RpmSclMin = 0$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
				Grenzwerte $> RpmMax$ und/oder Grenzwert $< RpmMin$ (nur wenn sie aktiviert sind) Oberer Grenzwert $<$ unterer Grenzwert
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 87 MonAnaDrv spezifischer Worst Quality Code

14.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable
MonEn	BOOL	FALSE	Enables dynamic and static Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s]
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s]
RpmAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
RpmAHLim	REAL	0	Limit Value for Alarm High
RpmALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
RpmALLim	REAL	100	Limit Value for Alarm Low
MonRpmTi	REAL	0.5	Monitor Time to recognize an Rpm limit error [s], only positive values

Tab. 88 MonAnaDrv persistente Variablen

14.2.5 Symbole

14.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Alarm symbols	 : Alarm ist aktiv	Siehe Beschreibung Funktionsbaustein und Alarmfunktion
Drive symbol	Grau: Motor ist im Stopp Blau: Motor läuft	
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	: kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Drehrichtung	 : Motor läuft vorwärts  : Motor läuft rückwärts	Bei Stopp wird kein Symbol angezeigt.
Rpm	Variable „RpmFbk“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	„RpmUnit“ wird dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles), Measurement Line (Nozzle mit durchgehender Linie) und Pipes (blaue Nozzles).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 89 MonAnaDrv Symbolbeschreibung

14.2.5.2 dMonAnaDrvPumpValue/uMonAnaDrvPumpValue/rMonAnaDrvPumpValue/lMonAnaDrvPumpValue

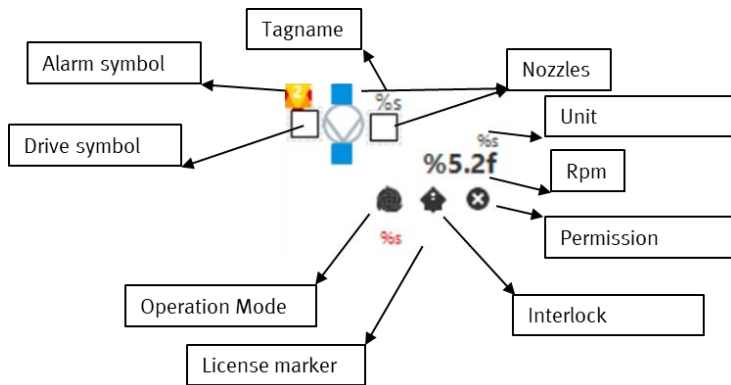


Fig. 92 Symbole MonAnaDrv Pump

14.2.5.3 hMonAnaDrvValue

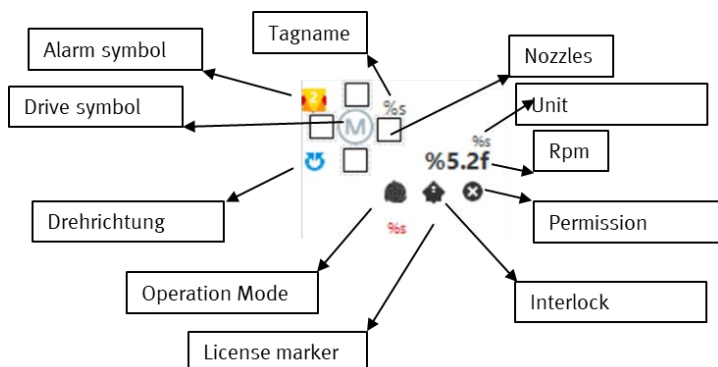


Fig. 93 Symbol hMonAnaDrvValue

14.2.6 Faceplates

14.2.6.1 Main

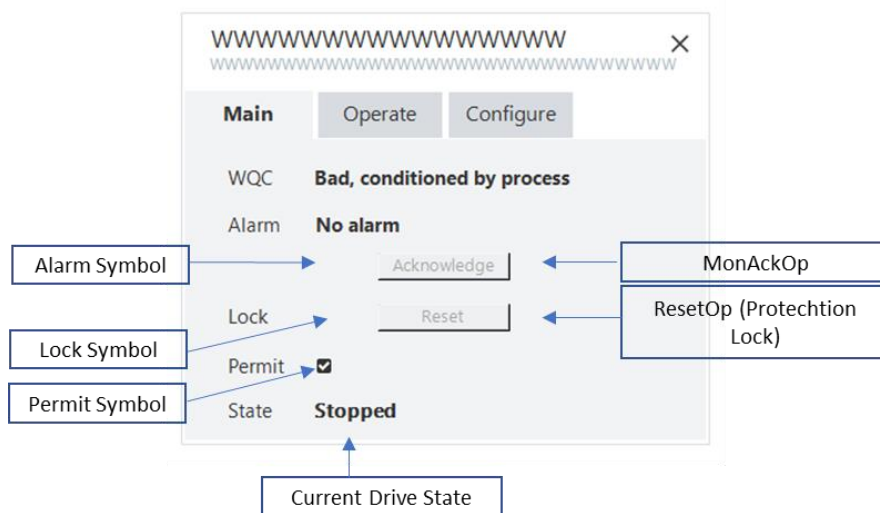


Fig. 94 MonAnaDrv Main Faceplate

14.2.6.2 Operate

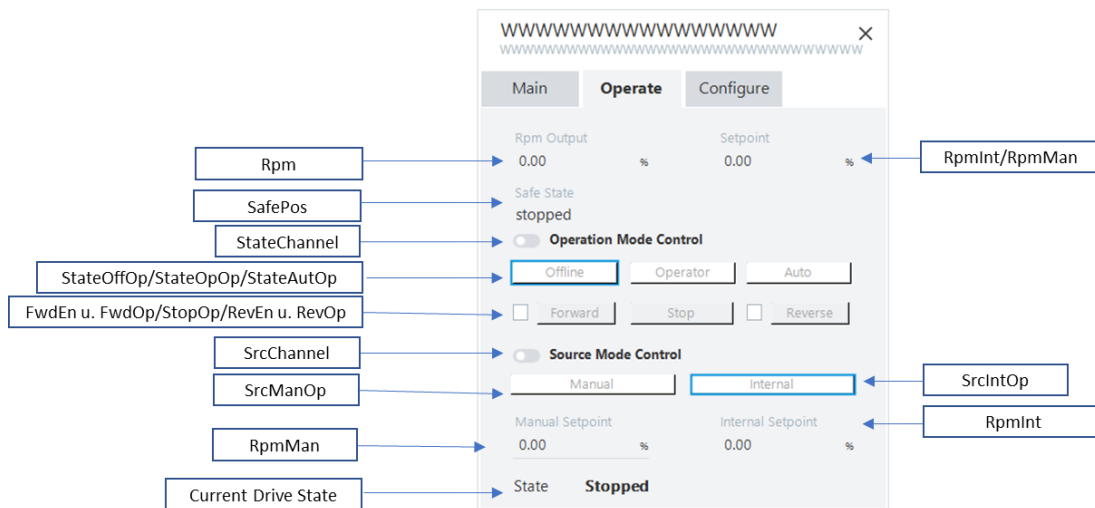


Fig. 95 MonAnaDrv Operate Faceplate

14.2.6.3 Configure

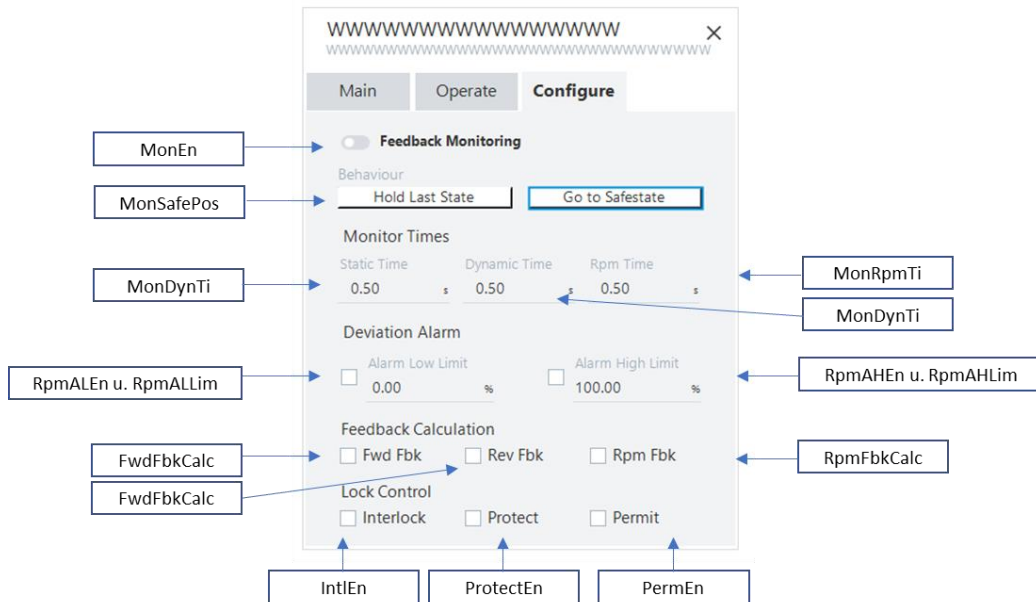


Fig. 96 MonAnaDrv Configure Faceplate

15 Totalizer

15.1 Funktionsbaustein Totalizer

15.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) Totalizer bietet die Möglichkeit einen Dosierprozess zu realisieren.

15.1.2 Grundfunktionen

15.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

15.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 90 Worst Quality Code Enumeration

15.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

15.1.3 Funktionen

15.1.3.1 Sollwertvorgabe „Splnt“ und „SpMan“

Der Sollwert für die zu dosierende Menge kann über zwei Quellen angegeben werden. Über den Eingang „Splnt“ kann direkt am FB der Sollwert vorgegeben werden. Soll der Sollwert über OPC UA oder das Faceplate eingestellt werden, so wird das Eingabefeld „Manual Setpoint“ genutzt. Welcher der beiden Eingänge gerade für den Dosierprozess herangezogen wird steuert der Source Mode. Bei der Initialisierung ist die interne Quelle und somit „Splnt“ aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erklärt.

15.1.3.2 Begrenzung Sollwert

Über die Eingänge am FB „SpMin“ und „SpMax“ können Grenzwerte für den Sollwert eingegeben werden. Ist der angegebene Sollwert („Splnt“ bzw. „SpMan“) größer „SpMax“ oder kleiner „SpMin“ so wird der Sollwert auf den entsprechenden Grenzwert begrenzt. Zusätzlich muss darauf geachtet werden, dass „SpMin“ nicht größer ist als „SpMax“. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

15.1.3.3 Einheit für „FlowIn“ und „TotalDos“

Für den Durchfluss „FlowIn“ kann die Einheit über den Eingang am FB „FlowUnit“ angegeben werden. Für die dosierte Menge „TotalDos“ kann die Einheit über den Eingang am FB „TotalDosUnit“ angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

15.1.3.4 Berechnung Durchfluss „FlowIn“

Der Eingang „FlowIn“ wird mit dem gemessenen Durchfluss verbunden. Dieser kann z.B. direkt von einem Durchflusssensor kommen. Für die Berechnung des aktuellen Durchflusses wird eine Zeitbasis zu „FlowIn“ benötigt. Diese wird über den Eingang am FB „FlowScl“ in Sekunden eingestellt und ist standardmäßig auf 60s eingestellt. Wird ein Wert kleiner 0 eingegeben, so wird zur Berechnung 60s angenommen. Die bereits dosierte Menge wird an den Ausgang „TotalDos“ ausgegeben.

15.1.3.5 Eingangsparameter „InFlight“

Über den Eingang am FB „InFlight“ lässt sich einstellen, wann das Ventil geschlossen werden soll. Dazu wird überprüft wann die dosierte Menge den Wert „Splnt/SpMan – InFlight“ erreicht. Das Ventil schließt somit bevor der eigentliche Sollwert erreicht wird. Bei großen Ventilen wird ein Überschreiten des Sollwertes dadurch vermieden. Werden Werte kleiner 0 eingegeben, so wird bei der internen Berechnung der Wert als 0 angesehen.

Zustände der Funktion InFlight:

Bedingungen	Operation Mode	Auswirkung
TotalDos > Sp - InFlight	Manual/Auto	Ventilzustand = 1
TotalDos = Sp - InFlight	Manual/Auto	Ventilzustand = 0
Setpoint < InFlight	Manual/Auto	InFlight = 0

Tab. 91 Zustände von InFlight

15.1.3.6 Cut-Off

Da bei einem geschlossenen Ventil der Wert von „FlowIn“ nie exakt 0 wird, kann über den Eingang am FB „CutOff“ der „Nullwert“ angegeben werden. Ist „FlowIn“ kleiner als „CutOff“, so gilt das Ventil als geschlossen. Das bedeutet auch, dass nach einem Start-Befehl erst eine Menge größer „CutOff“ beim Eingang „FlowIn“ als Durchfluss gesehen wird und zur Berechnung herangezogen wird. Werden Werte kleiner 0 eingegeben so wird bei der internen Berechnung der Wert als 0 angesehen.

Zustände der Funktion CutOff:

Bedingungen	Operation Mode	Auswirkung
$\text{CutOff} < 0$	Manual/Auto	Cutoff = 0
$\text{CutOff} > \text{FlowIn}$	Manual/Auto	Ventil = 0
$\text{CutOff} < \text{FlowIn} + \text{SP} > 0$	Manual/Auto	Ventil = 1

Tab. 92 Zustände von CutOff

15.1.3.7 Totalizersteuerung

Der Totalizer kann über die Steuerungskommandos Start, Stopp und Reset gesteuert werden. Mit Start wird das Ventil geöffnet und der Dosierprozess gestartet. „FlowIn“ muss größer „CutOff“ sein, damit der Wert berücksichtigt wird. Es wird solange dosiert, bis der Setpoint (unter Berücksichtigung von InFlight) erreicht wurde. Dann wird das Ventil geschlossen. Wenn „FlowIn“ kleiner als „CutOff“ ist, wird das Ventil als geschlossen angesehen und der Totalizer geht in den Zustand „Finished“.

Mit dem Steuerbefehl Stopp wird das Ventil geschlossen und der Abfüllprozess pausiert. Mit Reset wird der Abfüllprozess gestoppt und alle Werte werden wieder zurückgesetzt, so dass ein neuer Dosierprozess gestartet werden kann.

Ein Steuerbefehl wird abgegeben indem die oben genannten Variablen je nach Befehl auf TRUE gesetzt werden. Die Eingangsvariablen „StartAut“, „StopAut“ und „ResetAut“ sind flankengesteuert und reagieren auf ansteigende Flanken. „StartOp“, „StopOp“ und „ResetOp“ werden nach der Registrierung vom FB wieder auf FALSE gesetzt.

In der folgenden Tabelle ist aufgeführt wie sich der Ausgang VlvOut in Abhängigkeit der Steuerkommandos verhält.

Steuerkommando	Ausgänge
Stopp	VlvOut = 0 (Ventil geschlossen)
Start	VlvOut = 1
Reset	VlvOut = 0

Tab. 93 Steuerkommandos

Der FB kann in drei Zustände Offline, Operator und Automatic betrieben werden. Die Zustände werden über den Operation Mode gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Folgende Steuerbefehle und Ausgänge werden im jeweiligen Zustand verarbeitet und gesetzt:

- Offline Mode: keine Funktion Reset wird ausgeführt, VlvOut = 0
- Operator Mode: „StartOp“, „StopOp“, „ResetOp“, „SPMan“ oder „SPInt“ (über Faceplate änderbar)
- Automatic Mode: „StartAut“, „StopAut“, „ResetAut“, „SPMan“ oder „SPInt“ (Eingänge am FB)

15.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				FlowScl ≤ 0 ⇒ FlowScl = 60
				InFlight < 0 ⇒ InFlight = 0
				InFlight > Sp ⇒ InFlight = 0
				CutOff < 0 ⇒ CutOff = 0
				TotalDos < 0
				SpMin > SPMax
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 94 Totalizer spezifischer Worst Quality Code

15.1.5 Source Mode

15.1.5.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang SrcIntAct auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit SrcManAct auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

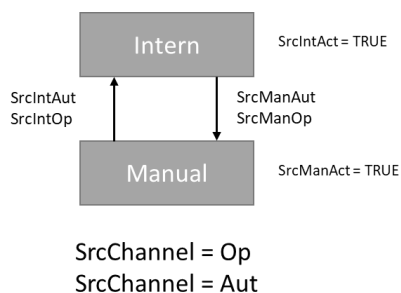


Fig. 97 Zustandsdiagramm Source Mode

15.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

15.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen SrcIntOp, SrcManOp, SrcIntAut oder SrcManAut an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

15.1.6 Operation Mode

15.1.6.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

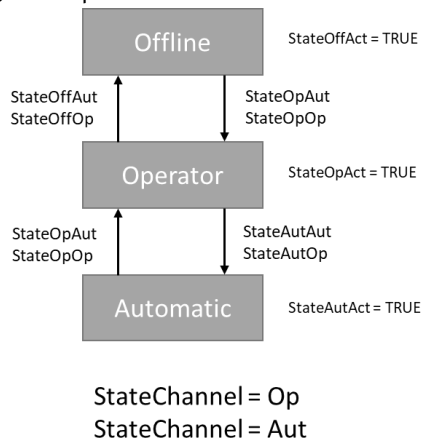


Fig. 98 Zustandsdiagramm Operation Mode

15.1.6.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

15.1.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

15.1.7 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
InFlight	REAL	0	Parameter to stop the dosing process before setpoint is reached (Splnt/SpMan - InFlight), only positive values
CutOff	REAL	0	No Flow Parameter (set the zero value for no Flow Input), only positive values
OSLevel	BYTE	0	OS Level Variable

Tab. 95 Totalizer persistente Variablen

15.1.8 Symbole

15.1.8.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Current setpoint	„Splnt“ oder „SpMan“ wird, in Abhängigkeit des Source Mode, dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Dosed amount	„TotalDos“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „TotalDosUnit“ wird beim Sollwert und der dosierten Menge dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Linie).	
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 96 Totalizer Symbolbeschreibung

15.1.8.2 hTotalizerValue

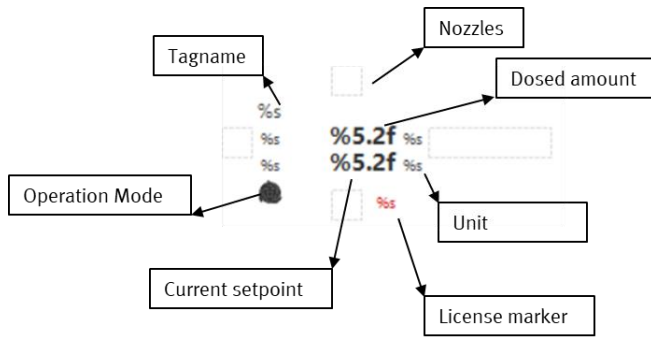


Fig. 99 Symbol hTotalizerValue

15.2 Funktionsbaustein MonTotalizer

15.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) MonTotalizer erweitert den FB Totalizer um die Grenzwertüberwachung.

15.2.2 Funktionen

15.2.2.1 Überprüfung der dosierten Menge „TotalDos“

Mit dieser Funktion wird am Ende eines Dosierprozess, d.h. wenn der Zustand „Finished erreicht ist (Sollwert erreicht, Ventil geschlossen und $\text{FlowIn} < \text{CutOff}$) die dosierte Menge „TotalDos“ überprüft. Es kann ein unterer prozentualer Toleranzwert „TolALLim“ und ein oberer prozentualer Toleranzwert „TolAHLim“ über Eingänge am FB eingestellt werden. Über diese prozentualen Werte und dem Setpoint wird ein Toleranzbereich berechnet. Nachdem der Dosierprozess beendet ist, wird überprüft, ob der Ausgang „TotalDos“ in dem berechneten Toleranzbereich liegt, andernfalls wird ein Alarm ausgegeben. Die Toleranzwerte können über Eingänge am FB („TolALEn“ und „TolAHEn“) einzeln aktiviert werden.

Werden Zahlen kleiner 0 als Toleranzwerte eingestellt so werden diese in positive Zahlen umgewandelt. Wird -5% eingegeben so wird der Wert auf 5% gesetzt.

Ein Alarm wird über das zugehörige Ausgangsbit ausgegeben („TolALerr“ und „TolAHErr“).

Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

15.2.2.2 Alarm quittieren

Liegt kein Fehler mehr vor, so werden die entsprechenden Ausgangsbits wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („TolAlmAckOp“) oder über den Eingang „TolAlmAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Sichtbarkeit des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 97 Zustände Alarmsymbol

Hinweis:

- Der Eingang „TolAlmAckAut“ ist flankengesteuert und reagiert nur auf ansteigende Flanken.

15.2.3 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode ist aktiv
				FlowScl <= 0 ⇒ FlowScl = 60
				InFlight < 0 ⇒ InFlight = 0
				InFlight > Sp ⇒ InFlight = 0
				CutOff < 0 ⇒ CutOff = 0
				TotalDos < 0
				SpMin > SPMaX
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 98 MonTotalizer spezifischer Worst Quality Code

15.2.4 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
InFlight	REAL	0	Parameter to stop the dosing process bevor setpoint is reached (SpInt/SpMan - InFlight), only positive values
CutOff	REAL	0	No Flow Parameter (set the zero value for no Flow Input), only positive values
OSLevel	BYTE	0	OS Level Variable
TotAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
TotAHLim	REAL	0	Limit Value for Alarm High [%]
TotALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
TotALLim	REAL	0	Limit Value for Alarm Low [%]

Tab. 99 MonTotalizer persistente Variablen

15.2.5 Symbole

15.2.5.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Alarm symbols	 : Alarm ist aktiv (höchste Priorität)	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Current setpoint	„Splnt“ oder „SpMan“ wird, in Abhängigkeit des Source Mode, dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Dosed amount	„TotalDos“ wird dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „TotalDosUnit“ wird beim Sollwert und der dosierten Menge dargestellt.	Nur Werte von Enumeration „UnitSettings“ möglich.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Linie).	Wichtig für den MTP-Exporter, damit Linien mit dem Symbol und anderen Linien richtig verbunden werden.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 100 MonTotalizer Symbolbeschreibung

15.2.5.2 hMonTotalizerValue

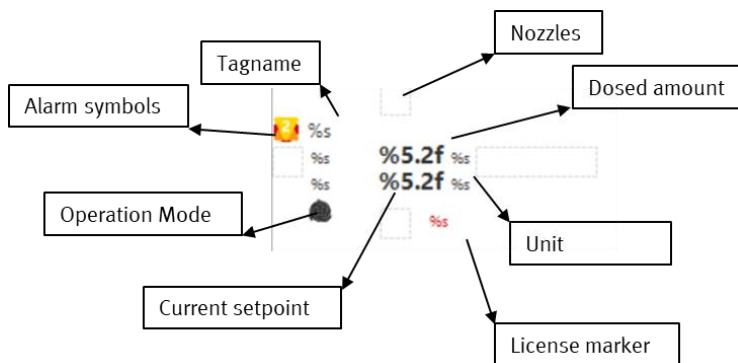


Fig. 100 Symbol hMonTotalizerValue

15.2.6 Faceplates

15.2.6.1 Main

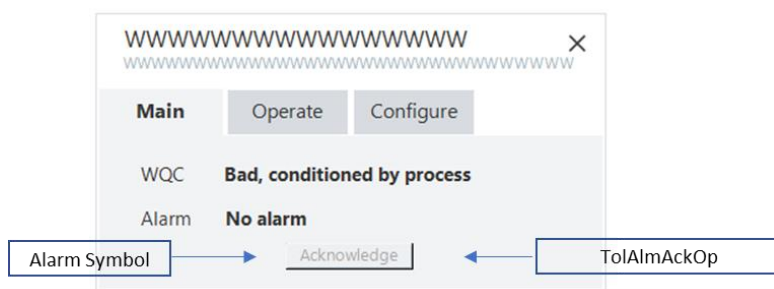


Fig. 101 MonTotalizer Main Faceplate

15.2.6.2 Operate

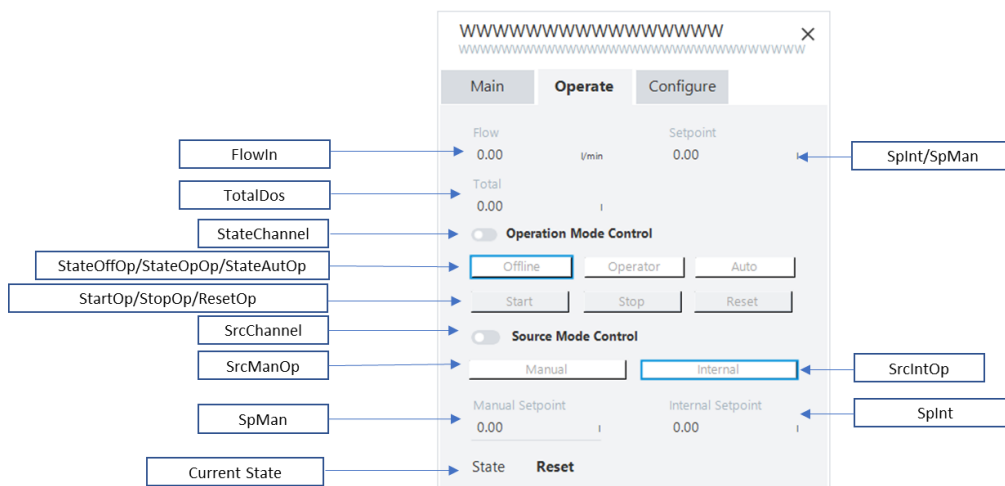


Fig. 102 MonTotalizer Operate Faceplate

15.2.6.3 Configure

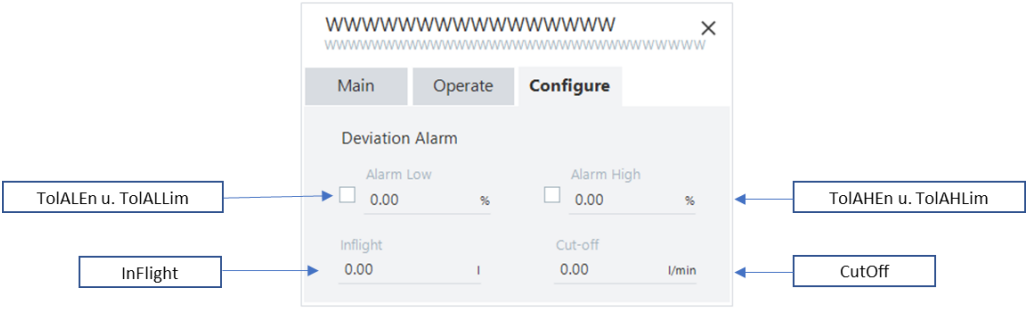


Fig. 103 MonTotalizer Configure Faceplate

16 MPA-Positioner

16.1 Funktionsbaustein DrvCtrl

16.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DrvCtrl bietet die Möglichkeit ein Dreh- oder Linearantrieb über Pilotventile zu betätigen und dessen Position zu erfassen. Zusammen mit den Ventilen und analogen Positionsrückmeldung wird die Funktion eines Positioners erfüllt.

Je nach Antriebsgröße, gewünschte Geschwindigkeit des Antriebs und Positioniergenauigkeit kann neben den Feinventil noch ein Grobventil angesteuert werden.

Der MPA-Positioner kann nicht verwendet werden, bis eine Kalibrierung durchgeführt wurde. Dabei wird der Antrieb in seine Endpositionen gefahren und die jeweiligen Positionsrückmeldungen gespeichert. Diese Werte bilden die Skala des Antriebs, von 0% bis 100%.

16.1.2 Grundfunktionen

16.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

16.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 101 Worst Quality Code Enumeration

16.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

16.1.3 Funktionen

16.1.3.1 Sollwertvorgabe „SpMan“ und „Splnt“

Der Sollwert für die Position kann über zwei Quellen angegeben werden. Über den Eingang „Splnt“ kann direkt am FB der Sollwert vorgegeben werden. Soll der Sollwert über OPC UA oder das Faceplate eingestellt werden, so wird das Eingabefeld „Manual Setpoint“ genutzt. Welcher der beiden Eingänge gerade für die Positionierung herangezogen wird steuert der Source Mode. Bei der Initialisierung ist die interne Quelle und somit „Splnt“ aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erklärt.

16.1.3.2 Begrenzung Sollwert

Die Begrenzung des Sollwerts ist nicht konfigurierbar und erfolgt durch die internen Variablen „SpMin“ und „SpMax“. „SpMin“ hat den Wert 0 und „SpMax“ den Wert 100.

Ist der angegeben Sollwert („Splnt“ bzw. „SpMan“) größer als „SpMax“ oder kleiner als „SpMin“, so wird der Sollwert auf den entsprechenden Grenzwert begrenzt.

16.1.3.3 Feedbacksignale Prozesswert (simuliert, tatsächlicher PV)

Der tatsächliche Prozesswert (PV) kann über den Eingang „FV“ am FB angebunden werden. Dieser wird dann in den eigentlichen „PV“ umgerechnet. Über den Eingang am FB „PVSimType“ kann der PV auch simuliert werden in dem ein Wert vorgegeben wird. Zusätzlich gibt es die Möglichkeit den PV auf den aktuellen Sollwert zu setzen.

16.1.3.4 Einheiten

Die Einheit wird, wie bei Positioner üblich, in % (Prozent) angegeben.

16.1.3.5 Sicherheitsposition – FailSafe

Bei einem Fehler oder einer Sperrung (z.B. Interlock) wird der Ausgang „FailSafeVlv“ auf FALSE gesetzt. Zusätzlich werden alle Ausgänge (GoTo0Vlv, GoTo100Vlv, CoarseGoTo0Vlv, Coarse GoTo100Vlv) auf den Wert 0 gesetzt.

16.1.3.6 Positionerfunktion

Der FB kann in drei Zustände Offline, Operator und Automatic betrieben werden. Die Zustände werden über den Operation Mode Control gesteuert. Dieser wird in einem extra Abschnitt beschrieben.

Folgende Steuerbefehle und Ausgänge werden im jeweiligen Zustand verarbeitet und gesetzt:

- Offline Mode: GoTo0Vlv = 0, GoTo100Vlv = 0, CoarseGoTo0Vlv = 0, Coarse GoTo100Vlv = 0, FailSafeVlv = 0
- Operator Mode: Move to 0% und Move to 100% (=TRUE) verfährt solange, bis Eingang wieder FALSE ist. Entsprechende Ventile werden angesteuert)
- Automatic Mode: In Abhängigkeit von PV und Sollwert werden Ventile gesteuert

16.1.3.7 Eingangsparmeter „UseCoarseVlv“

Über diesen Eingang lässt sich die Funktion für 'Coarse Valve (CV)' im Faceplate deaktivieren.

16.1.3.8 Kalibrierung

Über das Faceplate kann über den Button „Start“ in der Registerkarte „Calibrate“ oder direkt am FB über „CalibrateExt“ die Kalibrierung gestartet werden, indem die jeweilige Variable auf TRUE gesetzt wird.

16.1.3.9 Lock-Funktionen

16.1.3.9.1 Permit

Die Funktion Permit kann über den Eingang „PermEn“ (= TRUE) aktiviert werden. Ist der Eingang Permit TRUE, so können Steuerungsbefehle verarbeitet werden. Wird der Eingang „Permit“ auf FALSE gesetzt, so werden Steuerbefehle nicht mehr verarbeitet. Steuerkommandos (z.B. bei den Ventilen) werden nicht mehr ausgeführt. Ebenfalls werden keine Steuerausgänge gesetzt (z.B. bei den Operatoren). Der FB hält den letzten Zustand, es kann nur noch die Sicherheitsposition angefahren werden. Danach bleibt der FB in dieser Position bis der Eingang „Permit“ wieder auf TRUE gesetzt wird.

16.1.3.9.2 Interlock

Die Funktion Interlock kann über den Eingang „IntlEn“ (TRUE) aktiviert werden. Ist der Eingang „Interlock“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang „Interlock“ auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition hat, wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Es werden keine Steuerbefehle verarbeitet, bis der Eingang „Interlock“ wieder TRUE ist oder die Funktion über „IntlEn“ deaktiviert wird. Ist der Eingang „Interlock“ wieder TRUE so bleibt der FB im Sicherheitszustand bis ein neuer Steuerbefehl gesendet wird.

16.1.3.9.3 Protection Lock

Die Funktion Protection Lock kann über den Eingang „ProtEn“ (TRUE) aktiviert werden. Ist der Eingang „Protect“ gleich TRUE, so können Steuerbefehle verarbeitet werden. Wird der Eingang Protect auf FALSE gesetzt, so werden keine Steuerbefehle verarbeitet. Falls der FB eine Sicherheitsposition hat, wird dieser angefahren. Falls nicht wird der letzte Zustand gehalten. Wie beim Interlock werden auch hier keine Steuerbefehle verarbeitet, bis der Eingang „Protect“ wieder TRUE ist oder die Funktion über „ProtEn“ deaktiviert wurde. Zusätzlich muss der Protection Lock zurückgesetzt werden. Das kann in der Visualisierung über das Faceplate auf der Seite „Main“ über den Button „Reset“ erfolgen oder über den Eingang „ResetAut“ am FB. Achtung das zurücksetzen der Sperre ist erst wirksam, wenn der Eingang „Protect“ wieder TRUE ist.

Hinweis:

- Die Lock-Funktionalität ist in allen States (Offline, Operator, Auto) wirksam.
- „ResetAut“ ist flankengesteuert und reagiert auf ansteigende Flanken.

16.1.3.10 Positionsüberwachung

Über den Eingang am FB „DeadBand“ kann das Totband eingestellt werden. Dieses wird zum aktuellen Setpoint dazu addiert und subtrahiert. Dadurch entsteht ein Toleranzbereich. Verfährt der Positioner so muss in der Zeit „AlmDevTi“ der Toleranzbereich des Sollwerts erreicht werden, andernfalls wird ein Fehler ausgegeben. Ein Alarm wird über das zugehörige Ausgangsbit „AlmDevAct“ ausgegeben. Wird eine Visualisierung in CoDeSys verwendet, so wird der entsprechende Alarm im Symbol angezeigt.

Zahlen für den Parameter „DeadBand“ müssen positiv sein. Werden Zahlen kleiner 0 angegeben so werden diese in positive Zahlen umgewandelt. Wird -5 eingegeben so wird die Zeit auf 5s gesetzt.

Die Positionsüberwachung kann über den Eingang am FB „AlmDevEn“ (= TRUE) aktiviert werden.

16.1.3.11 Zeit „AlmDevTi“

Die Variable „AlmDevTi“ wird in Sekunden angegeben und ist standardmäßig auf 0.5s eingestellt. Diese Zeit kann über den Eingang am FB oder im Faceplate über das Feld „Alarm Time“ geändert werden. Zahlen kleiner 0 werden in positive Zahlen umgewandelt. Wird -5s eingegeben so wird die Zeit auf 5s gesetzt.

16.1.3.12 Alarm quittieren

Liegt keine Grenzwertverletzung mehr vor, so werden die entsprechenden Ausgangsbits „AlmDevAct“ wieder auf FALSE gesetzt. Im Symbol muss der Fehler aber von einem Operator über das Faceplate („Acknowledge“) oder über den Eingang „AlmDevAckAut“ am FB zurückgesetzt werden. In der Tabelle ist die Beschreibung des Alarmsymbols beschrieben.

Grenzwertverletzung noch da	Fehler quittiert	Alarm Symbol
Ja	Nein	Blinkt
Ja	Ja	Sichtbar
Nein	Nein	Blinkt
Nein	Ja	Unsichtbar-kein Fehler

Tab. 102 Zustände Alarmsymbol

Hinweis:

- Der Eingang „AlmDevAckAut“ ist flankengesteuert und reagiert nur auf steigende Flanken.
- Beim Quittieren werden alle aktuell anstehenden Alarmer gleichzeitig quittiert.
- Es können auch mehrere Grenzwertverletzungen gleichzeitig anstehen und über die entsprechenden Ausgangsbits ausgegeben werden – in der Visualisierung wird immer nur die höchste Priorität angezeigt.

16.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	Offline Mode is active
				SpInt/SpMan < SpMin ⇒ Sp = SpMin
				SpInt/SpMan > SpMax ⇒ Sp = SpMax
				SpMin > SpMax
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 103 DrvCtrl spezifischer Worst Quality Code

16.1.5 Source Mode

16.1.5.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (manual) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (*Int) so wird der Ausgang SrcIntAct auf TRUE gesetzt. Ist die manuelle Quelle (*Man) aktiv, so wird das Ausgangsbit SrcManAct auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

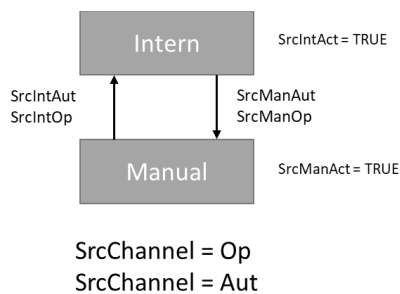


Fig. 104 Zustandsdiagramm Source Mode

16.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcManAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcManAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcManAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

16.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcManOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcManOp“ (manuelle Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcManOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcManAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcManOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen SrcIntOp, SrcManOp, SrcIntAut oder SrcManAut an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

16.1.6 Operation Mode

16.1.6.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Ein Zustandswechsel kann immer nur zwischen Offline und Operator oder Operator und Automatic erfolgen. Ein direkter Übergang von Offline in Automatic ist nicht erlaubt.

Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

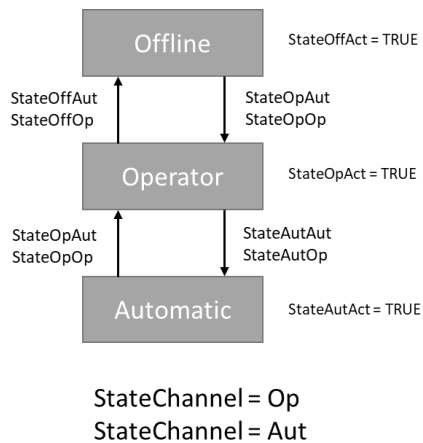


Fig. 105 Zustandsdiagramm Operation Mode

16.1.6.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

16.1.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

16.1.7 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
Scale100	REAL	20000	High Scale Value of Input (Scaling factor for PV calculation)
Scale0	REAL	4000	Low Scale Value of Field Value (Scaling factor for PV calculation)
AlmDevEn	BOOL	FALSE	Enables Deviation Alarm (1: enabled 0: disabled)
DeadBand	REAL	1.0	Deadband Limit Value (Inside the deadband the output is tracked, tolerance value for deviation alarm)
AlmDevTi	REAL	0.5	Monitor Time for Deviation Alarm [s]
Calibrated	BOOL	FALSE	Calibration has been finished
OSLevel	BYTE	0	OS Level Variable
CoarseThreshold	REAL	10.0	Full power with coarse valve over +ThresholdCoarse and under -ThresholdCoarse [%]
Threshold	REAL	5.0	Begin to pulse under +Threshold and above -Threshold [%]
PWMFreq	REAL	7.5	Sets the period for the pulsing sequence: 10 means 100ms
CyclesVlv100	ULINT	0,0	Number of cycles
CyclesVlv0	ULINT	0,0	Number of cycles
CyclesCoarseVlv100	ULINT	0,0	Number of cycles
CyclesCoarseVlv0	ULINT	0,0	Number of cycles
dtDateTimeUTC	DATE_AND_TIME	0,0	Save the date of last calibration
ValveOnOffTime	REAL	20.0	Time how long does the valve need to open the valve
CloseTightEnable	BOOL	TRUE	Enables CloseTight Parameter (1: enabled 0: disabled)
CloseTightThreshold	REAL	1.0	Parameter for Close Tight Threshold

Tab. 104 DrvCtrl persistente Variablen

16.1.8 Symbole

16.1.8.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Valve symbol	Grau: PV < 25 % Blau: PV > 25%	
Alarm symbol	 : Alarm ist aktiv (höchste Priorität)	Siehe Beschreibung Funktionsbaustein und Alarmfunktion.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Interlock	 : kein Interlock/Protection Lock aktiv  : Interlock ist aktiv  : Protection Lock ist aktiv	Siehe Beschreibung Lock-Funktion.
Permission	 : Zugriff zugelassen  : Zugriff verweigert	Siehe Beschreibung Lock-Funktion.
Setpoint	„Splnt“ oder „SpMan“ wird, in Abhängigkeit des Source Mode, dargestellt.	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Process Value	„PV“ wird dargestellt (tatsächlicher oder simulierter Wert).	Max. 5 Stellen vor dem Komma und 2 Stellen nach dem Komma werden angezeigt.
Unit	Einheit „%“ wird beim Sollwert und Prozesswert dargestellt.	Kann nicht verändert werden.
Nozzles	Nozzles zur Verbindung von Logical Lines (gestrichelte Nozzles) und Pipes (blaue Nozzles).	
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 105 DrvCtrl Symbolbeschreibung

16.1.8.2 hDrvCtrlValue/vDrvCtrlValue

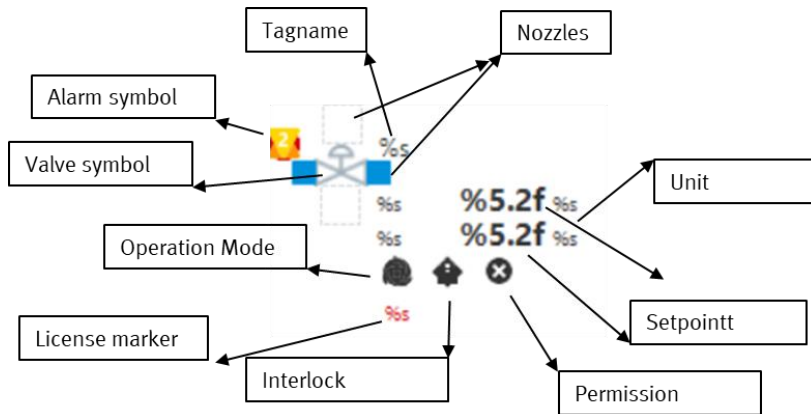


Fig. 106 Symbole hDrvCtrlValue und vDrvCtrlValue

16.1.8.3 hDrvCtrlLinear

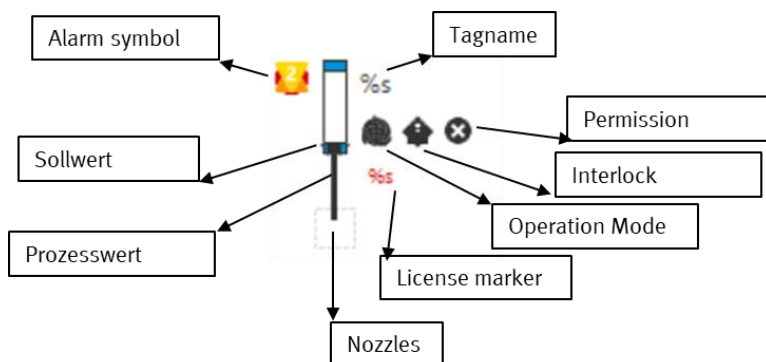


Fig. 107 Symbol hDrvCtrlLinear

16.1.9 Faceplates

16.1.9.1 Main

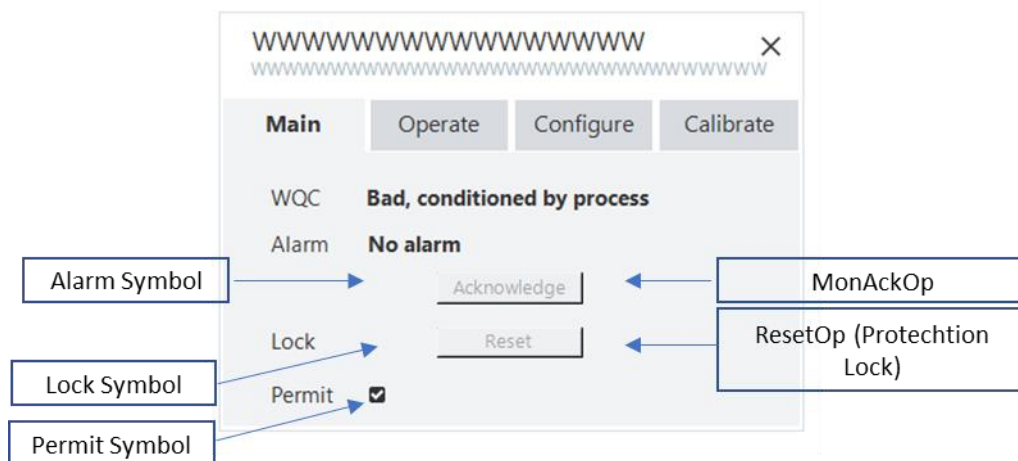


Fig. 108 DrvCtrl Main Faceplate

16.1.9.2 Operate

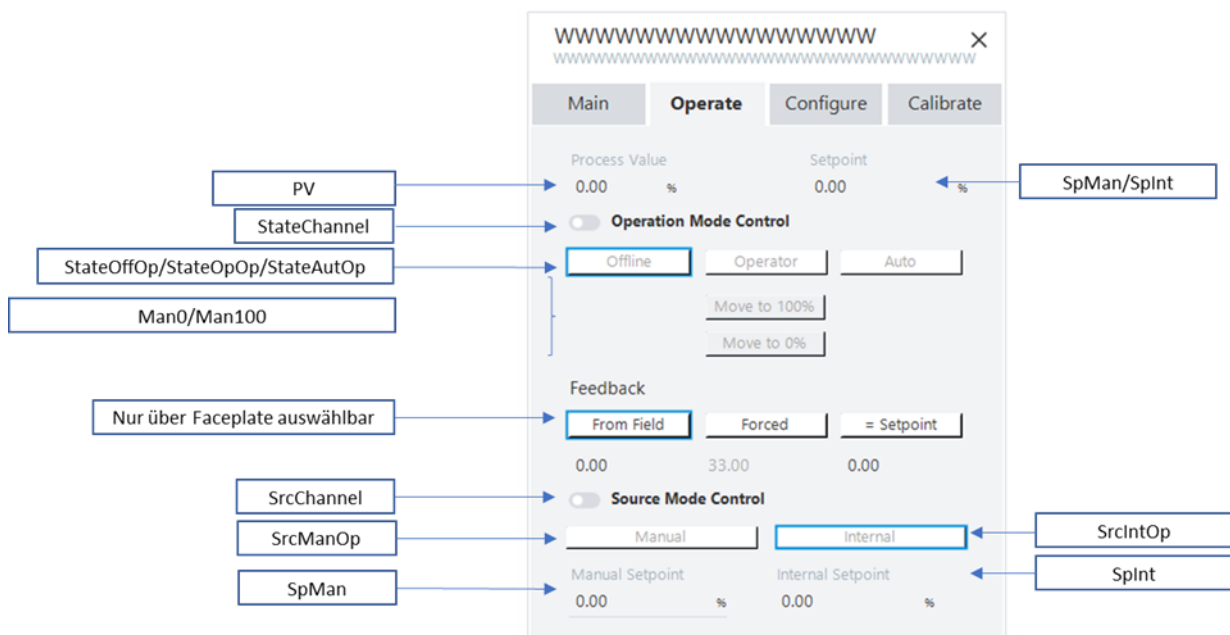


Fig. 109 DrvCtrl Operate Faceplate

16.1.9.3 Configure

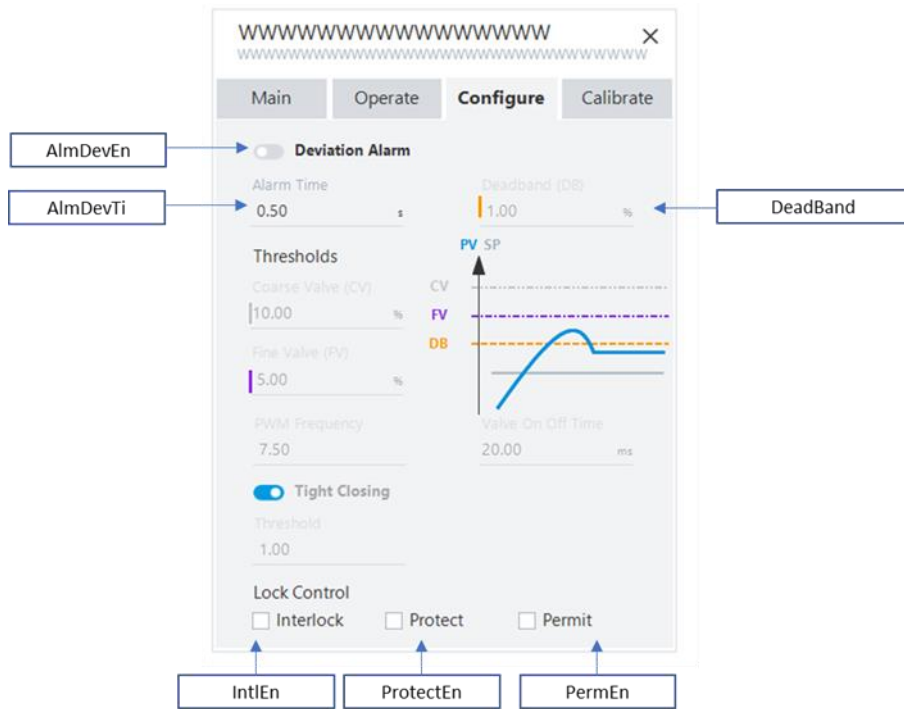


Fig. 110 DrvCtrl Configure Faceplate

16.1.9.4 Calibrate

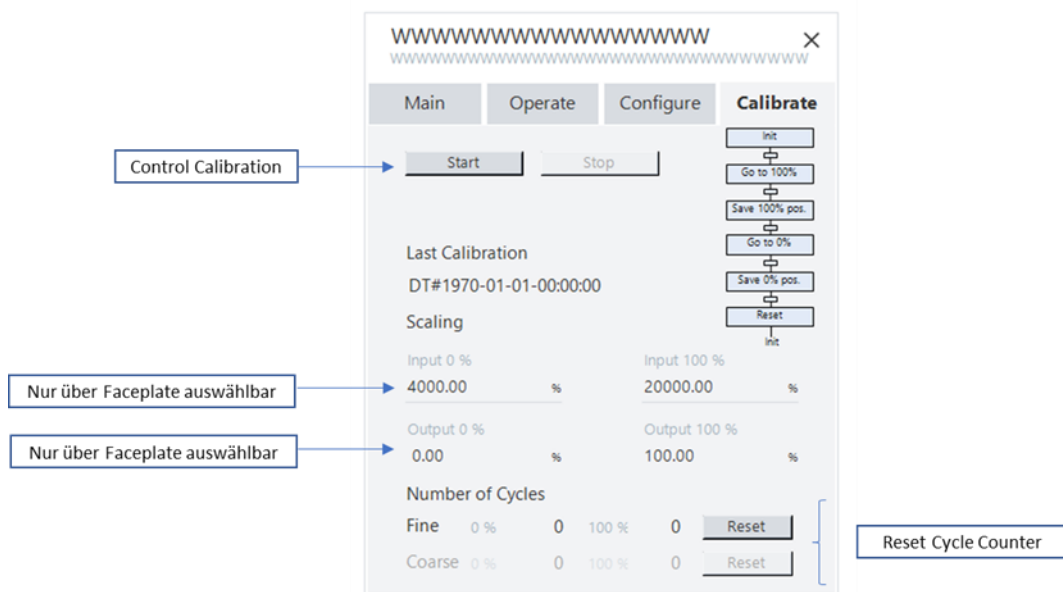


Fig. 111 DrvCtrl Calibrate Faceplate

17 Verriegelungsanzeige (Interlock)

17.1 Funktionsbaustein LockView4

17.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) LockView4 bietet die Möglichkeit eine Verriegelung mit Bedingungen zu realisieren. Das Ergebnis aus der Verknüpfung aller aktiven Eingänge wird dazu genutzt die Protect-, Interlock- und Permit-Eingänge an weiteren FBs zu setzen und somit Steuerbefehle zu sperren oder freizugeben.

17.1.2 Grundfunktionen

17.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

17.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 106 Worst Quality Code Enumeration

17.1.3 Funktionen

17.1.3.1 Eingangswerte „In1“, „In2“, „In3“, „In4“

Der FB hat 4 Eingänge die einzeln aktiviert (In1En, In2En...) werden können. Alle Werte der aktiven Eingänge werden für die logische Operation herangezogen und bilden den Ausgangswert.

17.1.3.2 Eingangswerte „In1QC“, „In2QC“, „In3QC“, „In4QC“

Jeder Eingang bekommt einen Worst Quality Code (WQC). Der WQC mit der höchsten Priorität wird an den Ausgang „QCOut“ ausgegeben.

17.1.3.3 Eingangswerte „In1Inv“, „In2Inv“, „In3Inv“, „In4Inv“

Werden diese Eingänge auf TRUE gesetzt so wird der Eingangswert (In1, In2, ...) als invertierter Wert für die logischen Operation verwendet.

17.1.3.4 Eingangswerte „In1Txt“, „In2Txt“, „In3Txt“, „In4Txt“

Für jeden Eingang kann ein Text mit 32 Zeichen für die Beschreibung des Eingangs hinterlegt werden.

17.1.3.5 Ausgangswert „Out“

Nach der logischen Operation wird der errechnete Wert an den Ausgang „Out“ ausgegeben. Der Worst Quality Code „QCOut“ gibt die Qualität des errechneten Wertes an. Sind alle Eingänge deaktiviert, so gibt der Ausgang „Out“ den Wert TRUE aus. „Out“ kann bei anderen FBs als Eingang für Permit, Protect und Interlock genutzt werden.

17.1.3.6 Logik

Über den Eingang am FB „Logic“ kann bestimmt werden, wie alle aktiven Eingänge verknüpft werden:

Logic 1: In1 AND In2 AND In3 AND In4

Wird ein aktivierter Eingang auf FALSE gesetzt, so wird „Out“ ebenfalls auf FALSE gesetzt und ein Permit, Interlock oder Protection Lock wird an dem verbundenen FB ausgelöst.

Logic 2: In1 OR In2 OR In3 OR In4

Erst wenn alle aktiven Eingänge auf FALSE gesetzt sind, so nimmt „Out“ auch den Wert FALSE an.

17.1.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 107 LockView4 spezifischer Worst Quality Code

17.1.5 Persistente Variablen

Bei diesem FB gibt es keine persistenten Variablen.

17.1.6 Symbole

17.1.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock symbol	Stellt den Zustand des Ausgangs dar.	Wenn „Out“ den Wert FLASE hat, wird das Symbol rot.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 108 LockView4 Symbolbeschreibung

17.1.6.2 hLockView4Default

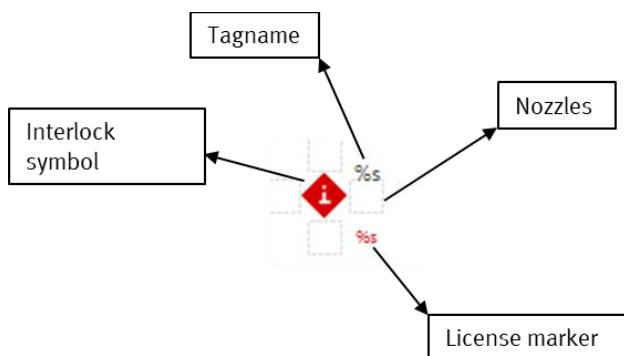


Fig. 112 Symbol hLockView4Default

17.1.7 Faceplates

17.1.7.1 Main

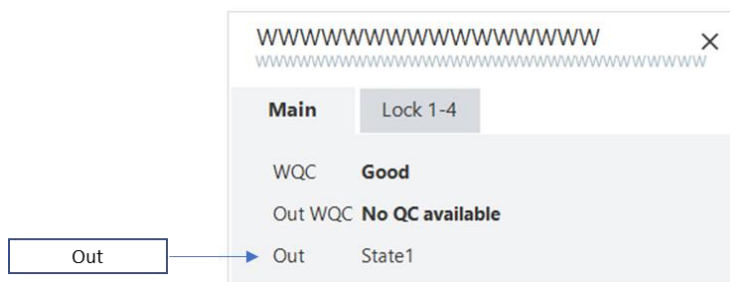


Fig. 113 LockView4 Main Faceplate

17.1.7.2 Lock 1-4

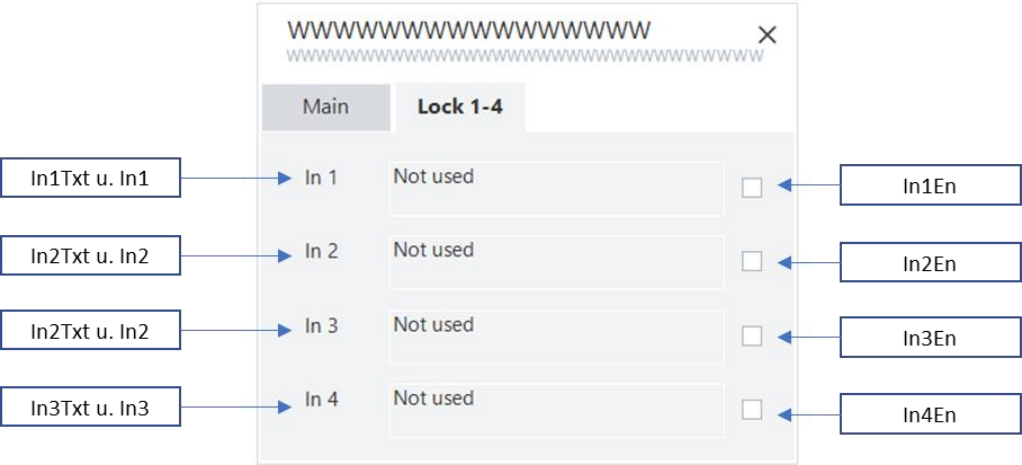


Fig. 114 LockView4 Lock1-4 Faceplate

17.2 Funktionsbaustein LockView8

17.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) LockView8 bietet die Möglichkeit eine Verriegelung mit Bedingungen zu realisieren. Das Ergebnis aus der Verknüpfung aller aktiven Eingänge wird dazu genutzt die Protect-, Interlock- und Permit-Eingänge an weiteren FBs zu setzen und somit Steuerbefehle zu sperren oder freizugeben.

17.2.2 Grundfunktionen

17.2.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

17.2.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 109 Worst Quality Code Enumeration

17.2.3 Funktionen

17.2.3.1 Eingangswerte „In1“, „In2“, „In3“, „In4“, ...

Der FB hat 8 Eingänge die einzeln aktiviert (In1En, In2En...) werden können. Alle Werte der aktiven Eingänge werden für die logische Operation herangezogen und bilden den Ausgangswert.

17.2.3.2 Eingangswerte „In1QC“, „In2QC“, „In3QC“, „In4QC“, ...

Jeder Eingang bekommt einen Worst Quality Code (WQC). Der WQC mit der höchsten Priorität wird an den Ausgang „QCOut“ ausgegeben.

17.2.3.3 Eingangswerte „In1Inv“, „In2Inv“, „In3Inv“, „In4Inv“, ...

Werden diese Eingänge auf TRUE gesetzt so wird der Eingangswert (In1, In2, ...) als invertierter Wert für die logischen Operation verwendet.

17.2.3.4 Eingangswerte „In1Txt“, „In2Txt“, „In3Txt“, „In4Txt“, ...

Für jeden Eingang kann ein Text mit 32 Zeichen für die Beschreibung des Eingangs hinterlegt werden.

17.2.3.5 Ausgangswert „Out“

Nach der logischen Operation wird der errechnete Wert an den Ausgang „Out“ ausgegeben. Der Worst Quality Code „QCOut“ gibt die Qualität des errechneten Wertes an. Sind alle Eingänge deaktiviert, so gibt der Ausgang „Out“ den Wert TRUE aus. „Out“ kann bei anderen FBs als Eingang für Permit, Protect und Interlock genutzt werden.

17.2.3.6 Logic

Über die Logic kann bestimmt werden, wie alle aktiven Eingänge verknüpft werden:

Logic 1: In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8

Wird ein aktivierter Eingang auf FALSE gesetzt, so wird „Out“ ebenfalls auf FALSE gesetzt und ein Permit, Interlock oder Protection Lock wird an dem verbundenen FB ausgelöst.

Logic 2: In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8

Erst wenn alle aktiven Eingänge auf FALSE gesetzt sind, so nimmt „Out“ auch den Wert FALSE an.

17.2.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 110 LockView8 spezifischer Worst Quality Code

17.2.5 Persistente Variablen

Bei diesem FB gibt es keine persistenten Variablen.

17.2.6 Symbole

17.2.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock symbol	Stellt den Zustand des Ausgangs dar.	Wenn „Out“ den Wert FLASE hat, wird das Symbol rot.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 111 LockView8 Symbolbeschreibung

17.2.6.2 hLockView8Default

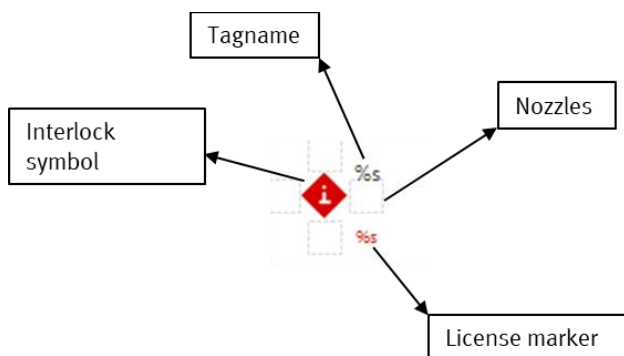


Fig. 115 hLockView8Default

17.2.7 Faceplates

17.2.7.1 Main

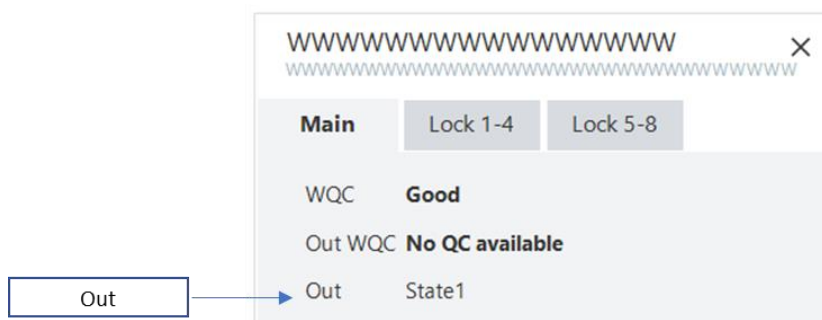


Fig. 116 LockView8 Main Faceplate

17.2.7.2 Lock 1-4



Fig. 117 LockView8 Lock1-4 Faceplate

17.2.7.3 Lock 5-8



Fig. 118 LockView8 Lock5-8 Faceplate

17.3 Funktionsbaustein LockView16

17.3.1 Kurzbeschreibung

Der Funktionsbaustein (FB) LockView16 bietet die Möglichkeit eine Verriegelung mit Bedingungen zu realisieren. Das Ergebnis aus der Verknüpfung aller aktiven Eingänge wird dazu genutzt die Protect-, Interlock- und Permit-Eingänge an weiteren FBs zu setzen und somit Steuerbefehle zu sperren oder freizugeben.

17.3.2 Grundfunktionen

17.3.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

17.3.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibung aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 112 Worst Quality Code Enumeration

17.3.3 Funktionen

17.3.3.1 Eingangswerte „In1“, „In2“, „In3“, „In4“, ...

Der FB hat 16 Eingänge die einzeln aktiviert (In1En, In2En...) werden können. Alle Werte der aktiven Eingänge werden für die logische Operation herangezogen und bilden den Ausgangswert.

17.3.3.2 Eingangswerte „In1QC“, „In2QC“, „In3QC“, „In4QC“, ...

Jeder Eingang bekommt einen Worst Quality Code (WQC). Der WQC mit der höchsten Priorität wird an den Ausgang „QCOut“ ausgegeben.

17.3.3.3 Eingangswerte „In1Inv“, „In2Inv“, „In3Inv“, „In4Inv“, ...

Werden diese Eingänge auf TRUE gesetzt so wird der Eingangswert (In1, In2, ...) als invertierter Wert für die logischen Operation verwendet.

17.3.3.4 Eingangswerte „In1Txt“, „In2Txt“, „In3Txt“, „In4Txt“, ...

Für jeden Eingang kann ein Text mit 32 Zeichen für die Beschreibung des Eingangs hinterlegt werden.

17.3.3.5 Ausgangswert „Out“

Nach der logischen Operation wird der errechnete Wert an den Ausgang „Out“ ausgegeben. Der Worst Quality Code „QCOut“ gibt die Qualität des errechneten Wertes an. Sind alle Eingänge deaktiviert, so gibt der Ausgang „Out“ den Wert TRUE aus. „Out“ kann bei anderen FBs als Eingang für Permit, Protect und Interlock genutzt werden.

17.3.3.6 Logic

Über die Logic kann bestimmt werden, wie alle aktiven Eingänge verknüpft werden:

Logic 1: In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8 AND In9 AND In10 AND In11 AND In12 AND In13 AND In14 AND In15 AND In16

Wird ein aktivierter Eingang auf FALSE gesetzt, so wird „Out“ ebenfalls auf FALSE gesetzt und ein Permit, Interlock oder Protection Lock wird an dem verbundenen FB ausgelöst.

Logic 2: In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8 OR In9 OR In10 OR In11 OR In12 OR In13 OR In14 OR In15 OR In16

Erst wenn alle aktiven Eingänge auf FALSE gesetzt sind, so nimmt „Out“ auch den Wert FALSE an.

17.3.4 FB-spezifischer Worst Quality Code

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 113 LockView16 spezifischer Worst Quality Code

17.3.5 Persistente Variablen

Bei diesem FB gibt es keine persistenten Variablen.

17.3.6 Symbole

17.3.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Interlock symbol	Stellt den Zustand des Ausgangs dar.	Wenn „Out“ den Wert FLASE hat, wird das Symbol rot.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 114 LockView16 Symbolbeschreibung

17.3.6.2 hLockView16Default

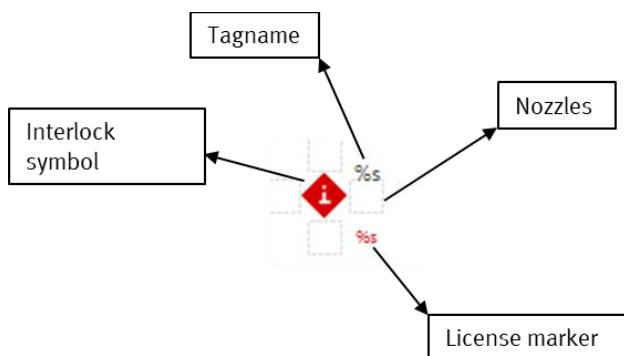


Fig. 119 Symbol hLockView16Default

17.3.7 Faceplates

17.3.7.1 Main

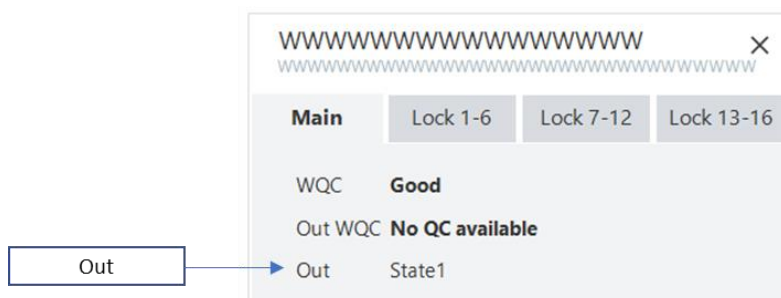


Fig. 120 LockView16 Main Faceplate

17.3.7.2 Lock 1-6

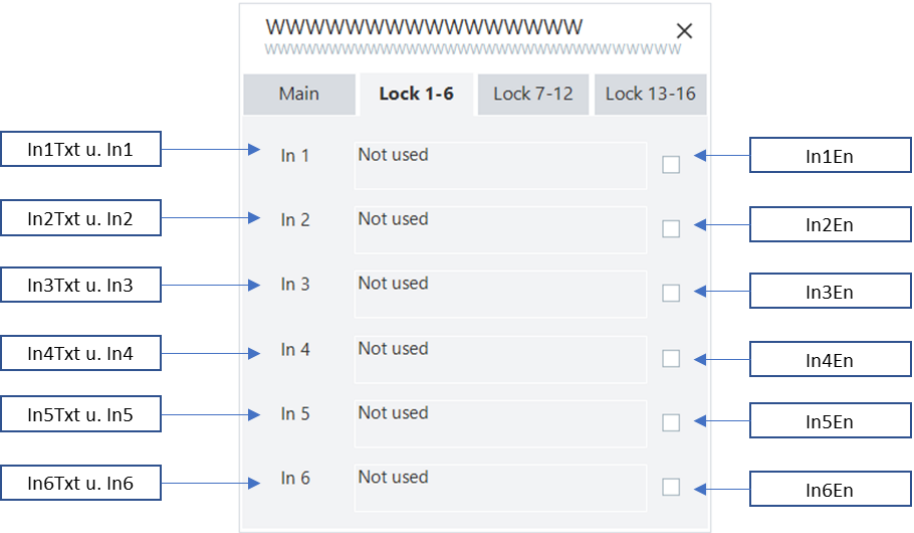


Fig. 121 LockView Lock1-6 Faceplate

17.3.7.3 Lock 7-12

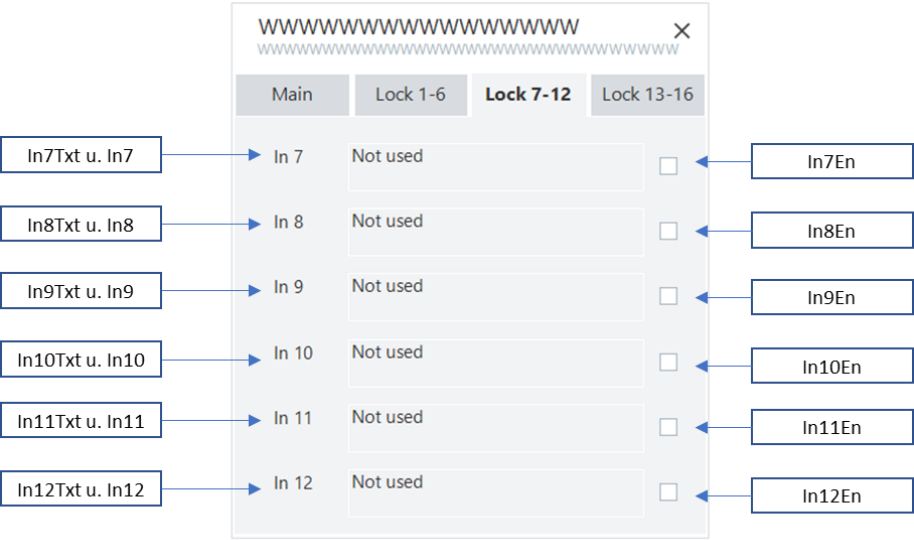


Fig. 122 LockView16 Lock7-12 Faceplate

17.3.7.4 Lock 13-16

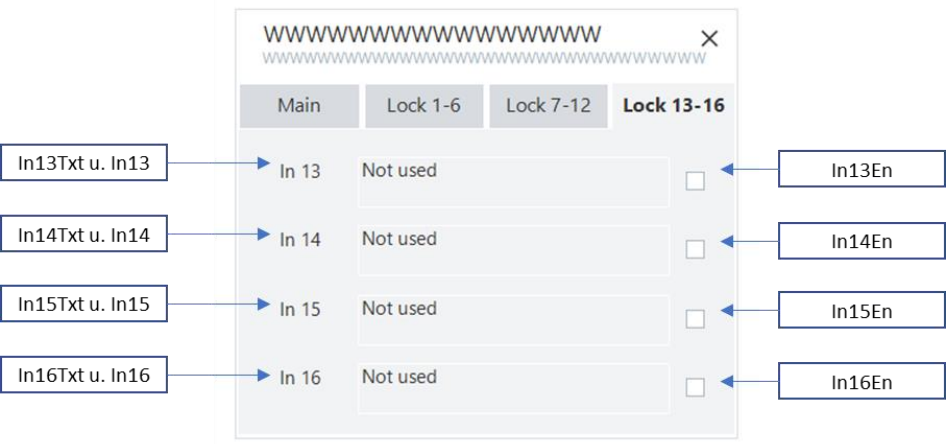


Fig. 123 LockView16 Lock13-16 Faceplate

18 Services

18.1 SimpleSelfCompletingServiceControl

18.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) SimpleSelfCompletingServiceControl ermöglicht die Implementierung eines selbstbeendenden Dienstes gemäß einer standardisierten Zustandsmaschine (s. Abbildung). Der Funktionsbaustein gibt hierbei den Rahmen vor, die konkrete Funktionalität innerhalb eines der 16 Zustände kann applikationsabhängig implementiert werden. Der Übergang zwischen den 16 Zuständen der Zustandsmaschine erfolgt entweder automatisch (bei transienten Zuständen – in der Abbildung gestrichelt dargestellt) oder durch ein vorgegebenes Steuerkommando (bei nichttransienten Zuständen – in der Abbildung durchgängig dargestellt).

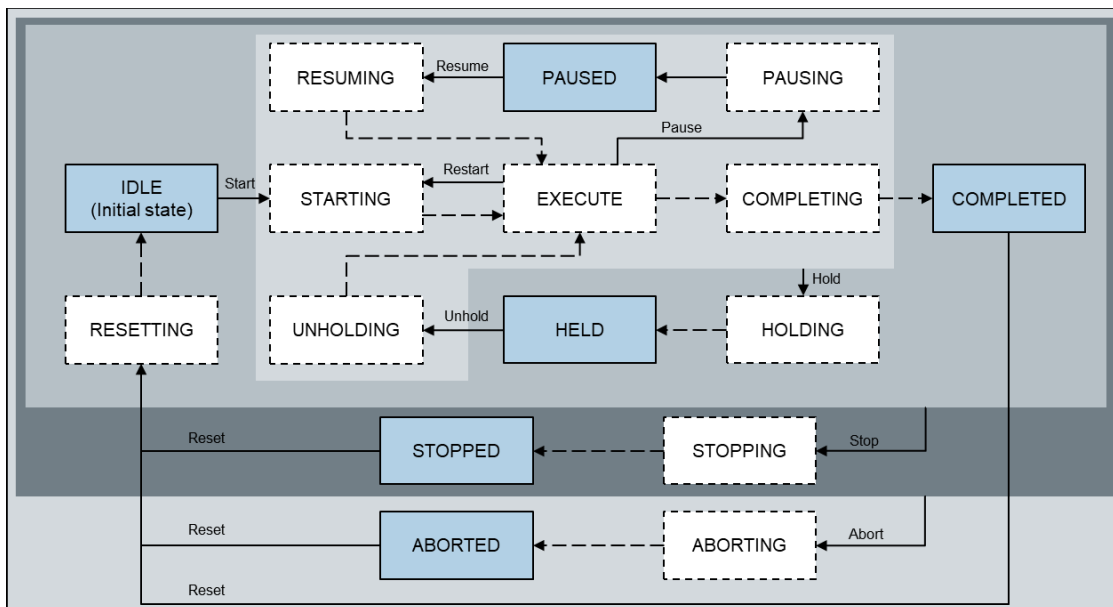


Fig. 124 Zustandsdiagramm SimpleSelfCompletingServiceControl

Ein Dienst selbst besitzt drei Zustände – Offline, Operator und Automatic. Im Zustand Offline ist der Dienst nicht betriebsbereit. Demgegenüber ist der Dienst in den beiden Zuständen Operator und Automatic aktiv, d.h. obige Zustandsmaschine wird ausgeführt.

Im Zustand Operator wird ein Kommando zum Auslösen eines Zustandsübergangs über die Variable CommandOp durch den Operator vorgegeben. Der Zustand Automatic ist bspw. für die Nutzung des Dienstes im Rahmen einer Rezeptabarbeitung gedacht – in diesem Fall wird entweder ein intern oder extern vorgegebenes Kommando genutzt. Die Schnittstelle beinhaltet hierzu eine entsprechende Umschaltung, wobei die Funktion Source Mode zum Einsatz kommt.

18.1.2 Grundfunktionen

18.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

18.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration			
Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 115 Worst Quality Code Enumeration

18.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

18.1.3 Funktionen

18.1.3.1 Implementierung und Nutzung applikationsspezifischer Dienste

Der FB SimpleSelfCompletingServiceControl stellt das Grundgerüst für einen anzubietenden Dienst dar inklusive obiger Zustandsmaschine bereit. Die Funktionalität, die innerhalb der verschiedenen Zustände durchgeführt werden soll, ist aber applikationsspezifisch. Soll ein solcher applikationsspezifischer Dienst angeboten werden, ist daher zunächst ein entsprechender FB anzulegen, der den FB SimpleSelfCompletingServiceControl erweitert:

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Beispieldienst EXTENDS SimpleSelfCompletingServiceControl  
(* Beispielimplementierung ST *)  
SUPER^0;
```

Für diesen applikationsspezifischen FB kann anschließend die Funktionalität der benötigten Zustände definiert werden. Jeder Zustand wird dazu als Aktion vom Typ Ablaufsprache (AS) implementiert.

Hinweis:

- ⇒ Die verfügbaren Zustände stehen bei ‚Rechtsklick -> Objekt hinzufügen -> Aktion...‘ automatisch zur Auswahl.
- ⇒ Nicht benötigte Zustände müssen nicht implementiert werden.

Achtung:

Damit die Grundimplementierung des Bausteins ServiceControl den aktuellen Zustand der Schrittketten der jeweiligen Zustände abfragen und (beim Verlassen des jeweiligen Zustands) zurücksetzen kann, muss der Zugriff auf die AS-Flags SFCInit und SFCCurrentStep aktiviert werden. Dies kann Projekt-weit in den Projekteinstellungen (Projekt -> Projekteinstellungen -> AS) wie folgt aktiviert werden:

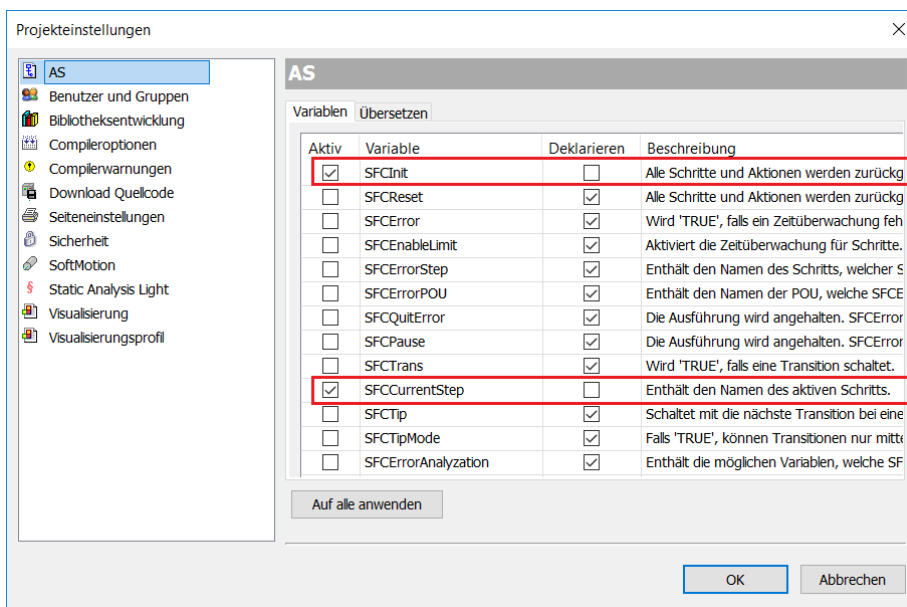


Fig. 125 Projekteinstellung

Alternativ kann diese Einstellung auch für jeden implementierten Zustand einzeln (in den Eigenschaften der Zustands-Aktion) eingestellt werden (Rechtsklick -> Eigenschaften... -> AS-Einstellungen):

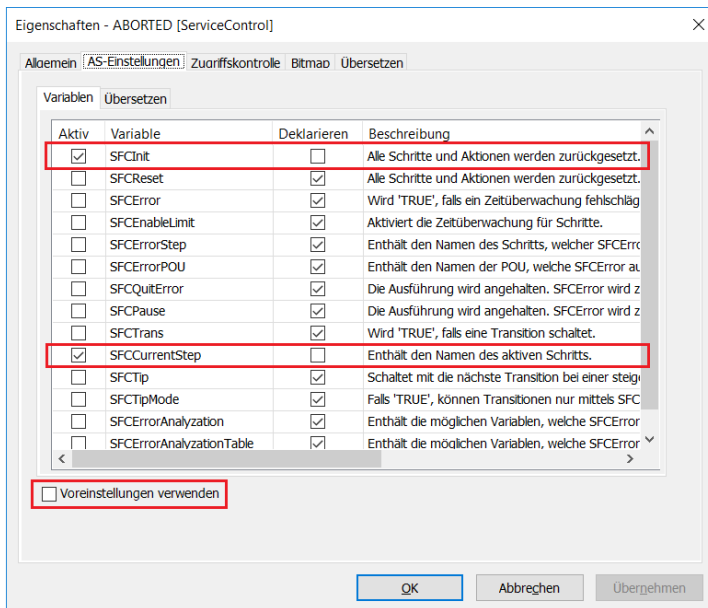


Fig. 126 Individuelle Projekteinstellungen

Anschließend kann der abgeleitete FB wie gewohnt instanziiert und genutzt werden:

```
(* Beispieldeklaration *)
VAR
    Instanz_Beispieldienst: Beispieldienst;
END_VAR
(* Beispielimplementierung ST *)
Instanz_Beispieldienst();
```

18.1.3.2 Transiente und nicht transiente Zustände

Transiente Zustände (in obiger Abbildung gestrichelt dargestellt) führen nach einmaliger Abarbeitung, d.h. sobald wieder in den Initialschritt der zugeordneten AS-Implementierung gesprungen wird, automatisch zu einem Übergang in den darauffolgenden Zustand. Demgegenüber werden nichttransiente Zustände (in obiger Abbildung durchgängig dargestellt) erst dann verlassen, wenn ein entsprechendes Kommando bspw. durch den Operator gesendet wird. Folglich kann in einem nichttransienten Zustand auch eine Schleife implementiert werden, die mehrfach den Initialzustand durchläuft.

18.1.3.3 Verhindern von Zustandsübergängen

Grundsätzlich sind Zustandsübergänge zwischen den 16 Zuständen gemäß obiger Abbildung aktiv. In einigen Fällen kann es jedoch sinnvoll sein, bestimmte Übergänge temporär zu blockieren (bspw. um das gleichzeitige Starten zweier Dienste zu verhindern, die auf die gleichen Aktoren zugreifen). Hierzu stehen die entsprechenden Eingänge Prevent...Command bereit.

18.1.3.4 Anzeige des aktuellen (Teil-Schrittes) in der Visualisierung

Im Faceplate eines Dienstes wird grundsätzlich der gerade aktive Zustand angezeigt (bspw. Execute). Da in einem Zustand jedoch komplexe Algorithmen mit mehreren Teilschritten ablaufen können, kann es sinnvoll sein, dem Operator über das Faceplate genauere Informationen über den aktuellen Bearbeitungsschritt mitzuteilen. Hierzu kann über den Eingang ActualStep eine beliebige, maximal 24 Zeichen lange Zeichenkette angegeben werden, die dann zusätzlich im Faceplate angezeigt wird. Werden die hierzu verwendeten Texte in einer Textliste verwaltet, die auch einem übergeordneten System (bspw. einem Rezeptmanager) vorliegen, kann zusätzlich über den Eingang PosTextID die ID des gerade angezeigten Textes angegeben werden. Da diese Variable über OPC UA gelesen werden kann, ermöglicht dieser Mechanismus die Anzeige des aktuellen Bearbeitungsschrittes auch in einem solchen übergeordneten System.

18.1.3.5 Möglichkeit für Rückfragen an den Bediener

Um Rückfragen an einen Bediener in Form eines einfachen Dialogs mit vordefinierten Fragen und Antwortmöglichkeiten zu realisieren, dient der Eingang InteractQuestionID. Über diesen kann die ID einer Frage aus einer Textliste spezifiziert werden. Der Bediener kann anschließend eine Antwort geben, in dem er einer von mehreren vordefinierten Antwortmöglichkeiten auswählt. Die Antwort kann dann bspw. im Rahmen der Implementierung eines Zustands des Service über die interne Variable InteractAnswerID ausgewertet werden.

18.1.3.6 Vorbereiten bzw. freigeben verwendeter Sensoren/Aktoren

Ein Service wird in den allermeisten Fällen auf einen oder mehrere Sensoren und/oder Aktoren (bspw. Instanzen der FBs AnaView oder AnaDrv) zugreifen. Daher ist sicherzustellen, dass diese Sensoren und Aktoren in einem entsprechenden nutzbaren Zustand sind, bevor der Service vom Zustand Offline in Operator oder Automatic geschaltet wird. Hierzu definiert der Baustein die Methode regainActuatorControls, die im Rahmen eines entsprechenden Zustandsübergangs aufgerufen wird. Wird diese Methode überschrieben, können hier speziellen Algorithmen implementiert werden, um bspw. alle genutzten Sensoren/Aktoren ebenfalls in den Zustand Operator bzw. Automatic zu schalten. Äquivalent steht die Methode releaseActuatorControls bereit, um genutzte Sensoren/Aktoren falls notwendig freizugeben, sobald ein Service deaktiviert und in den Zustand Offline überführt wird.

Hinweis:

- Die beiden Methoden stehen bei ‚Rechtsklick -> Objekt hinzufügen -> Methode...‘ automatisch zur Auswahl.

18.1.4 Erweiterte Funktionen

18.1.4.1 Nutzung von Prozeduren

Der Baustein SimpleSelfCompletingServiceControl erweitert den Baustein ServiceControl und besitzt daher auch die Möglichkeit der Definition und Auswahl unterschiedlicher Prozeduren. Für tiefergehende Erläuterungen zu dieser Thematik sei auf die Dokumentation der Bausteine ServiceControl und ServiceProcedure verwiesen. Intern stellt der Baustein bereits eine (Default-)Prozedur mit der ID 1 bereit, die standardmäßig aktiviert wird, solange keine weitere Prozedur definiert wird.

18.1.5 Source Mode

18.1.5.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

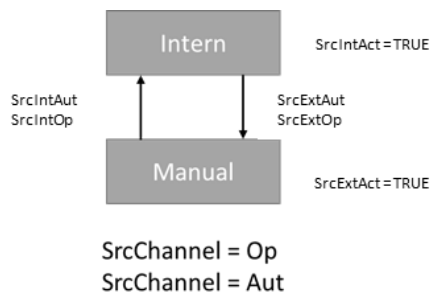


Fig. 127 Zustandsdiagramm Source Mode

18.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

18.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

18.1.6 Operation Mode

18.1.6.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variablen durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

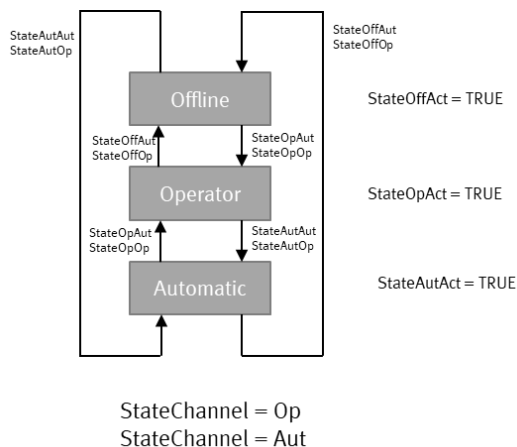


Fig. 128 Zustandsdiagramm Operation Mode

18.1.6.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

18.1.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.

- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

18.2 SimpleContinuousServiceControl

18.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) SimpleContinuousServiceControl ermöglicht die Implementierung eines kontinuierlichen Dienstes gemäß einer standardisierten Zustandsmaschine (s. Abbildung). Der Funktionsbaustein gibt hierbei den Rahmen vor, die konkrete Funktionalität innerhalb eines jeden der 16 Zustände kann applikationsabhängig implementiert werden. Der Übergang zwischen den 16 Zuständen der Zustandsmaschine erfolgt entweder automatisch (bei transienten Zuständen – in der Abbildung gestrichelt dargestellt) oder durch ein vorgegebenes Steuerkommando (bei nichttransienten Zuständen – in der Abbildung durchgängig dargestellt).

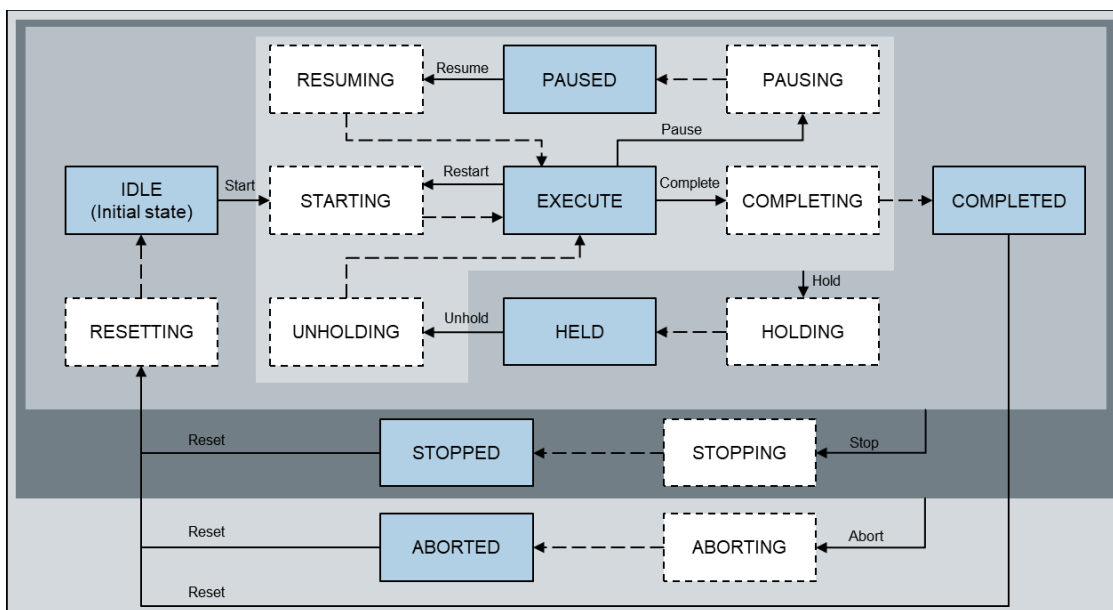


Fig. 129 Zustandsdiagramm SimpleContinuousServiceControl

Ein Dienst selbst besitzt drei Zustände – Offline, Operator und Automatic. Im Zustand Offline ist der Dienst nicht betriebsbereit. Demgegenüber ist der Dienst in den beiden Zuständen Operator und Automatic aktiv, d.h. obige Zustandsmaschine wird ausgeführt.

Im Zustand Operator wird ein Kommando zum Auslösen eines Zustandsübergangs über die Variable CommandOp durch den Operator vorgegeben. Der Zustand Automatic ist bspw. für die Nutzung des Dienstes im Rahmen einer Rezeptabarbeitung gedacht – in diesem Fall wird entweder ein intern oder extern vorgegebenes Kommando genutzt. Die Schnittstelle beinhaltet hierzu eine entsprechende Umschaltung, wobei die Funktion Source Mode zum Einsatz kommt.

18.2.2 Grundfunktionen

18.2.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

18.2.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration			
Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 116 Worst Quality Code Enumeration

18.2.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

18.2.3 Funktionen

18.2.3.1 Implementierung und Nutzung applikationsspezifischer Dienste

Der FB SimpleSelfCompletingServiceControl stellt das Grundgerüst für einen anzubietenden Dienst dar inklusive obiger Zustandsmaschine bereit. Die Funktionalität, die innerhalb der verschiedenen Zustände durchgeführt werden soll, ist aber applikationsspezifisch. Soll ein solcher applikationsspezifischer Dienst angeboten werden, ist daher zunächst ein entsprechender FB anzulegen, der den FB SimpleSelfCompletingServiceControl erweitert:

(* Beispieldeklaration *)

FUNCTION_BLOCK Beispieldienst EXTENDS SimpleContinuousServiceControl

(* Beispielimplementierung ST *)

SUPER^();

Für diesen applikationsspezifischen FB kann anschließend die Funktionalität der benötigten Zustände definiert werden. Jeder Zustand wird dazu als Aktion vom Typ Ablaufsprache (AS) implementiert.

Hinweis:

- Die verfügbaren Zustände stehen bei ‚Rechtsklick -> Objekt hinzufügen -> Aktion...‘ automatisch zur Auswahl.
- Nicht benötigte Zustände müssen nicht implementiert werden.

Achtung:

Damit die Grundimplementierung des Bausteins ServiceControl den aktuellen Zustand der Schrittketten der jeweiligen Zustände abfragen und (beim Verlassen des jeweiligen Zustands) zurücksetzen kann, muss der Zugriff auf die AS-Flags SFCInit und SFCCurrentStep aktiviert werden. Dies kann Projekt-weit in den Projekteinstellungen (Projekt -> Projekteinstellungen -> AS) wie folgt aktiviert werden:

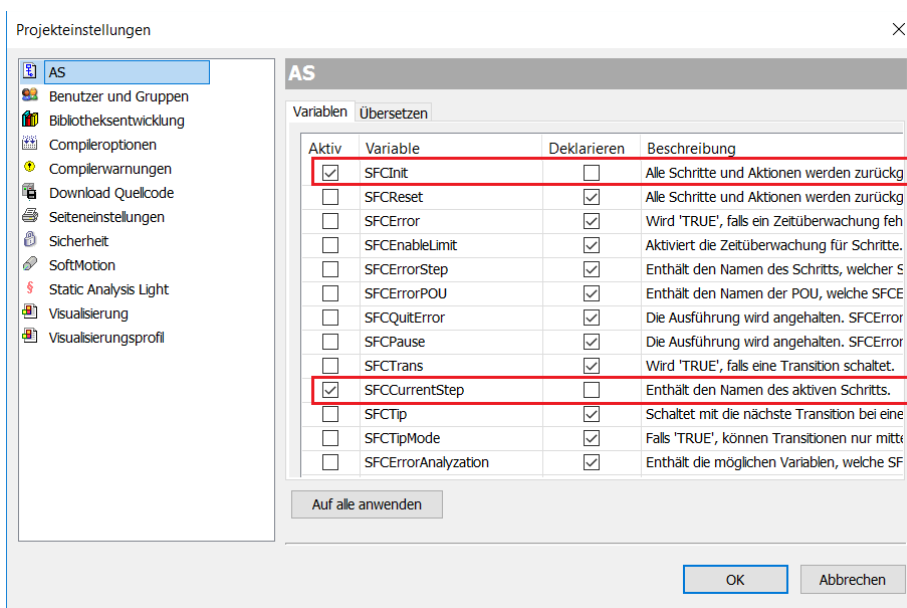


Fig. 130 Projekteinstellungen

Alternativ kann diese Einstellung auch für jeden implementierten Zustand einzeln (in den Eigenschaften der Zustands-Aktion) eingestellt werden (Rechtsklick -> Eigenschaften... -> AS-Einstellungen):

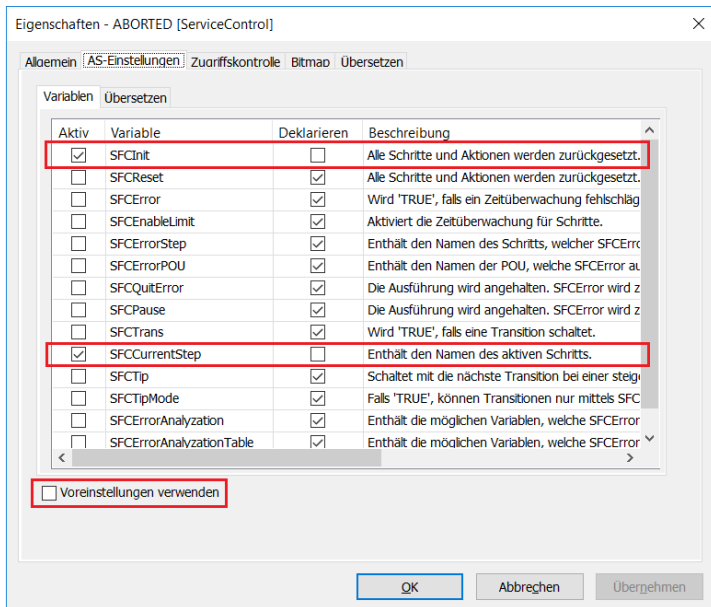


Fig. 131 Individuelle Projekteinstellungen

Anschließend kann der abgeleitete FB wie gewohnt instanziiert und genutzt werden:

```
(* Beispieldeklaration *)
VAR
    Instanz_Beispieldienst : Beispieldienst;
END_VAR
(* Beispielimplementierung ST *)
Instanz_Beispieldienst();
```

18.2.3.2 Transiente und nicht transiente Zustände

Transiente Zustände (in obiger Abbildung gestrichelt dargestellt) führen nach einmaliger Abarbeitung, d.h. sobald wieder in den Initialschritt der zugeordneten AS-Implementierung gesprungen wird, automatisch zu einem Übergang in den darauffolgenden Zustand. Demgegenüber werden nichttransiente Zustände (in obiger Abbildung durchgängig dargestellt) erst dann verlassen, wenn ein entsprechendes Kommando bspw. durch den Operator gesendet wird. Folglich kann in einem nichttransienten Zustand auch eine Schleife implementiert werden, die mehrfach den Initialzustand durchläuft.

18.2.3.3 Verhindern von Zustandsübergängen

Grundsätzlich sind Zustandsübergänge zwischen den 16 Zuständen gemäß obiger Abbildung aktiv. In einigen Fällen kann es jedoch sinnvoll sein, bestimmte Übergänge temporär zu blockieren (bspw. um das gleichzeitige Starten zweier Dienste zu verhindern, die auf die gleichen Aktoren zugreifen). Hierzu stehen die entsprechenden Eingänge Prevent...Command bereit.

18.2.3.4 Anzeige des aktuellen (Teil-Schrittes) in der Visualisierung

Im Faceplate eines Dienstes wird grundsätzlich der gerade aktive Zustand angezeigt (bspw. Execute). Da in einem Zustand jedoch komplexe Algorithmen mit mehreren Teilschritten ablaufen können, kann es sinnvoll sein, dem Operator über das Faceplate genauere Informationen über den aktuellen Bearbeitungsschritt mitzuteilen. Hierzu kann über den Eingang ActualStep eine beliebige, maximal 24 Zeichen lange Zeichenkette angegeben werden, die dann zusätzlich im Faceplate angezeigt wird. Werden die hierzu verwendeten Texte in einer Textliste verwaltet, die auch einem übergeordneten System (bspw. einem Rezeptmanager) vorliegen, kann zusätzlich über den Eingang PosTextID die ID des gerade angezeigten Textes angegeben werden. Da diese Variable über OPC UA gelesen werden kann, ermöglicht dieser Mechanismus die Anzeige des aktuellen Bearbeitungsschrittes auch in einem solchen übergeordneten System.

18.2.3.5 Möglichkeit für Rückfragen an den Bediener

Um Rückfragen an einen Bediener in Form eines einfachen Dialogs mit vordefinierten Fragen und Antwortmöglichkeiten zu realisieren, dient der Eingang InteractQuestionID. Über diesen kann die ID einer Frage aus einer Textliste spezifiziert werden. Der Bediener kann anschließend eine Antwort geben, in dem er einer von mehreren vordefinierten Antwortmöglichkeiten auswählt. Die Antwort kann dann bspw. im Rahmen der Implementierung eines Zustands des Service über die interne Variable InteractAnswerID ausgewertet werden.

18.2.3.6 Vorbereitung bzw. freigeben verwendeter Sensoren/Aktoren

Ein Service wird in den allermeisten Fällen auf einen oder mehrere Sensoren und/oder Aktoren (bspw. Instanzen der FBs AnaView oder AnaDrv) zugreifen. Daher ist sicherzustellen, dass diese Sensoren und Aktoren in einem entsprechenden nutzbaren Zustand sind, bevor der Service vom Zustand Offline in Operator oder Automatic geschaltet wird. Hierzu definiert der Baustein die Methode regainActuatorControls, die im Rahmen eines entsprechenden Zustandsübergangs aufgerufen wird. Wird diese Methode überschrieben, können hier speziellen Algorithmen implementiert werden, um bspw. alle genutzten Sensoren/Aktoren ebenfalls in den Zustand Operator bzw. Automatic zu schalten. Äquivalent steht die Methode releaseActuatorControls bereit, um genutzte Sensoren/Aktoren falls notwendig freizugeben, sobald ein Service deaktiviert und in den Zustand Offline überführt wird.

Hinweis:

- Die beiden Methoden stehen bei ‚Rechtsklick -> Objekt hinzufügen -> Methode...‘ automatisch zur Auswahl.

18.2.4 Erweiterte Funktionen

18.2.4.1 Nutzung von Prozeduren

Der Baustein SimpleSelfCompletingServiceControl erweitert den Baustein ServiceControl und besitzt daher auch die Möglichkeit der Definition und Auswahl unterschiedlicher Prozeduren. Für tiefergehende Erläuterungen zu dieser Thematik sei auf die Dokumentation der Bausteine ServiceControl und ServiceProcedure verwiesen. Intern stellt der Baustein bereits eine (Default-)Prozedur mit der ID 1 bereit, die standardmäßig aktiviert wird, solange keine weitere Prozedur definiert wird.

18.2.5 Source Mode

18.2.5.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

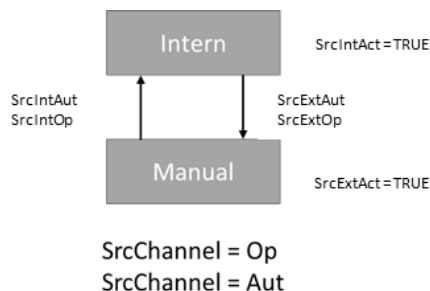


Fig. 132 Zustandsdiagramm Source Mode

18.2.5.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

18.2.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

18.2.6 Operation Mode

18.2.6.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variablen durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

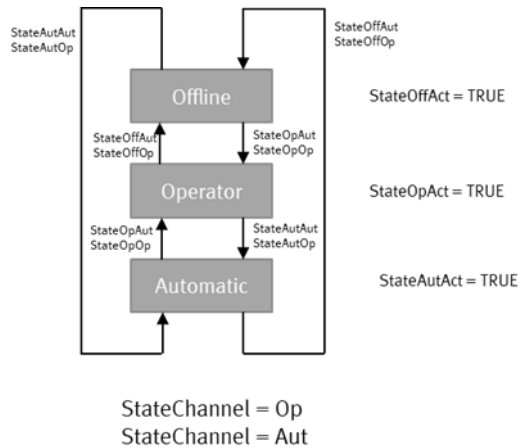


Fig. 133 Zustandsdiagramm Operation Mode

18.2.6.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

18.2.6.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.

- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

18.3 ServiceControl

18.3.1 Kurzbeschreibung

Der Funktionsbaustein (FB) ServiceControl ermöglicht die Implementierung eines komplexen Dienstes gemäß einer standardisierten Zustandsmaschine (s. Abbildung). Der Funktionsbaustein gibt hierbei den Rahmen vor, die konkrete Funktionalität innerhalb eines jeden der 16 Zustände kann applikationsabhängig implementiert werden. Der Übergang zwischen den 16 Zuständen der Zustandsmaschine erfolgt entweder automatisch (bei transienten Zuständen – in der Abbildung gestrichelt dargestellt) oder durch ein vorgegebenes Steuerkommando (bei nichttransienten Zuständen – in der Abbildung durchgängig dargestellt).

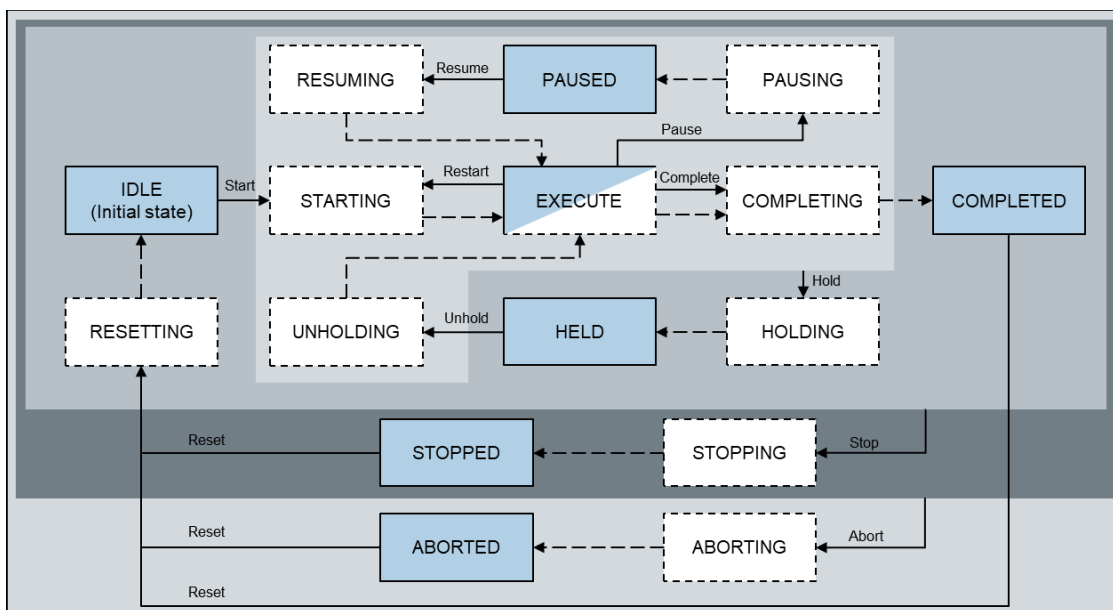


Fig. 134 Zustandsdiagramm ServiceControl

Ein Dienst selbst besitzt drei Zustände – Offline, Operator und Automatic. Im Zustand Offline ist der Dienst nicht betriebsbereit. Demgegenüber ist der Dienst in den beiden Zuständen Operator und Automatic aktiv, d.h. obige Zustandsmaschine wird ausgeführt.

Im Zustand Operator wird ein Kommando zum Auslösen eines Zustandsübergangs über die Variable CommandOp durch den Operator vorgegeben. Der Zustand Automatic ist bspw. für die Nutzung des Dienstes im Rahmen einer Rezeptarbeitung gedacht – in diesem Fall wird entweder ein intern oder extern vorgegebenes Kommando genutzt. Die Schnittstelle beinhaltet hierzu eine entsprechende Umschaltung, wobei die Funktion Source Mode zum Einsatz kommt.

Ein Dienst kann über mehrere Prozeduren bzw. Fahrweisen verfügen, die unterschiedliche Ausprägungen dieses Dienstes repräsentieren. Zur Auswahl einer zu aktivierenden Prozedur stehen analog zum Auslösen eines Kommandos die drei Variablen ProcedureOp (im Zustand Operator) bzw. ProcedureInt/Ext (im Zustand Automatic) bereit.

Zu beachten ist, dass eine Prozedur lediglich im Zustand Starting aktiviert wird. Ändert sich der Wert der aktiven Variable zur Auswahl der Prozedur bspw. im Zustand Execute, wird diese erst nach einem Neustart des Dienstes (und einem entsprechenden Durchlaufen des Starting-Zustands) aktiviert.

18.3.2 Grundfunktionen

18.3.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

18.3.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 117 Worst Quality Code Enumeration

18.3.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

18.3.3 Funktionen

18.3.3.1 Implementierung und Nutzung applikationsspezifischer Dienste

Der FB ServiceControl stellt das Grundgerüst für einen anzubietenden Dienst dar inklusive obiger Zustandsmaschine bereit. Die Funktionalität, die innerhalb der verschiedenen Zustände durchgeführt werden soll, ist aber applikationsspezifisch. Soll ein solcher applikationsspezifischer Dienst angeboten werden, ist daher zunächst ein entsprechender FB anzulegen, der den FB ServiceControl erweitert:

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Beispieldienst EXTENDS ServiceControl  
(* Beispielimplementierung ST *)  
SUPER^0;
```

Für diesen applikationsspezifischen FB kann anschließend die Funktionalität der benötigten Zustände definiert werden. Jeder Zustand wird dazu als Aktion vom Typ Ablaufsprache (AS) implementiert.

Hinweis:

- Die verfügbaren Zustände stehen bei ‚Rechtsklick -> Objekt hinzufügen -> Aktion...‘ automatisch zur Auswahl.
- Nicht benötigte Zustände müssen nicht implementiert werden.

Achtung:

Damit die Grundimplementierung des Bausteins ServiceControl den aktuellen Zustand der Schrittketten der jeweiligen Zustände abfragen und (beim Verlassen des jeweiligen Zustands) zurücksetzen kann, muss der Zugriff auf die AS-Flags SFCInit und SFCCurrentStep aktiviert werden. Dies kann Projekt-weit in den Projekteinstellungen (Projekt -> Projekteinstellungen -> AS) wie folgt aktiviert werden:

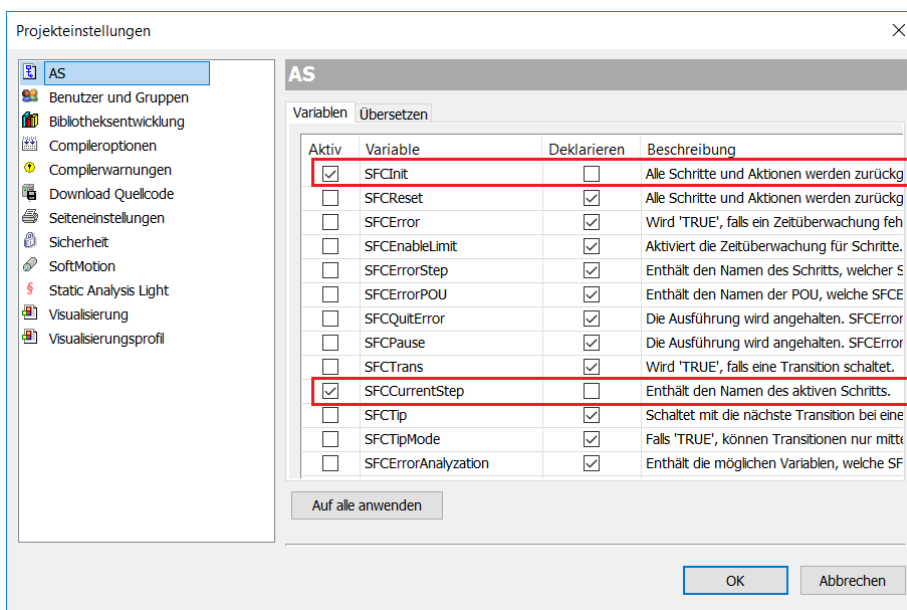


Fig. 135 Projekteinstellungen

Alternativ kann diese Einstellung auch für jeden implementierten Zustand einzeln (in den Eigenschaften der Zustands-Aktion) eingestellt werden (Rechtsklick -> Eigenschaften... -> AS-Einstellungen):

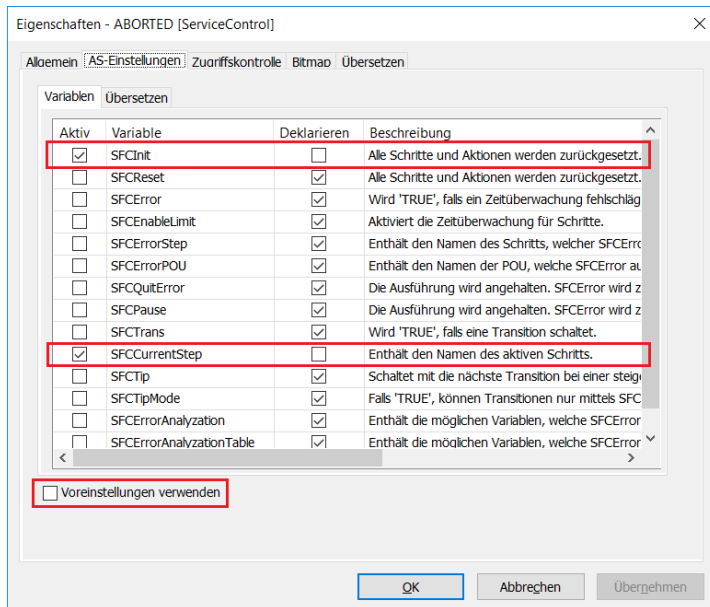


Fig. 136 Individuelle Projekteinstellungen

Abschließend kann der abgeleitete FB wie gewohnt instanziiert und genutzt werden:

(* Beispieldeklaration *)

VAR

Instanz_Beispieldienst : Beispieldienst;

END_VAR

(* Beispielimplementierung ST *)

Instanz_Beispieldienst();

18.3.3.2 Spezifikation und Auswahl von Prozeduren

Jeder Dienst kann beliebig viele Prozeduren definieren und hierzu den FB ServiceProcedure instanziiieren (für weitere Informationen sei auf die Dokumentation dieses FBs verwiesen). Zwingend erforderlich ist jedoch die Bereitstellung einer Default-Prozedur mit der ID 1. Zusätzlich sind zwingend die beiden Methoden isProcedureIDValid und isProcedureContinuous zu überschreiben. IsProcedureIDValid zeigt dem Dienst an, ob eine bestimmte angeforderte Prozedur existiert. Ist dies nicht der Fall, schaltet der Dienst automatisch auf die Default-Prozedur (mit der ID 1) um. Die Methode isProcedureContinuous gibt demgegenüber an, ob eine Prozedur selbstbeendend oder kontinuierlich ist und steuert und anderem die Freigabe bzw. das Auslösen des Übergangs zwischen den Zuständen Excecute und Completing.

Hinweis:

- Die beiden Methoden stehen bei, Rechtsklick -> Objekt hinzufügen -> Methode...'automatisch zur Auswahl.

18.3.3.3 Transiente und nicht transiente Zustände

Transiente Zustände (in obiger Abbildung gestrichelt dargestellt) führen nach einmaliger Abarbeitung, d.h. sobald wieder in den Initialschritt der zugeordneten AS-Implementierung gesprungen wird, automatisch zu einem Übergang in den darauffolgenden Zustand. Demgegenüber werden nichttransiente Zustände (in obiger Abbildung durchgängig dargestellt) erst dann verlassen, wenn ein entsprechendes Kommando bspw. durch den Operator gesendet wird. Folglich kann in einem nichttransienten Zustand auch eine Schleife implementiert werden, die mehrfach den Initialzustand durchläuft.

Ein Sonderfall stellt hierbei der Zustand Execute dar, der je nach Typ der gerade aktiven Prozedur entweder kontinuierlich oder selbstbeendend sein kann.

18.3.3.4 Verhindern von Zustandsübergängen

Grundsätzlich sind Zustandsübergänge zwischen den 16 Zuständen gemäß obiger Abbildung aktiv. In einigen Fällen kann es jedoch sinnvoll sein, bestimmte Übergänge temporär zu blockieren (bspw. um das gleichzeitige Starten zweier Dienste zu verhindern, die auf die gleichen Aktoren zugreifen). Hierzu stehen die entsprechenden Eingänge Prevent...Command bereit.

18.3.3.5 Anzeige des aktuellen (Teil-Schrittes) in der Visualisierung

Im Faceplate eines Dienstes wird grundsätzlich der gerade aktive Zustand angezeigt (bspw. Execute). Da in einem Zustand jedoch komplexe Algorithmen mit mehreren Teilschritten ablaufen können, kann es sinnvoll sein, dem Operator über das Faceplate genauere Informationen über den aktuellen Bearbeitungsschritt mitzuteilen. Hierzu kann über den Eingang ActualStep eine beliebige, maximal 24 Zeichen lange Zeichenkette angegeben werden, die dann zusätzlich im Faceplate angezeigt wird. Werden die hierzu verwendeten Texte in einer Textliste verwaltet, die auch einem übergeordneten System (bspw. einem Rezeptmanager) vorliegen, kann zusätzlich über den Eingang PosTextID die ID des gerade angezeigten Textes angegeben werden. Da diese Variable über OPC UA gelesen werden kann, ermöglicht dieser Mechanismus die Anzeige des aktuellen Bearbeitungsschrittes auch in einem solchen übergeordneten System.

18.3.3.6 Möglichkeit für Rückfragen an den Bediener

Um Rückfragen an einen Bediener in Form eines einfachen Dialogs mit vordefinierten Fragen und Antwortmöglichkeiten zu realisieren, dient der Eingang InteractQuestionID. Über diesen kann die ID einer Frage aus einer Textliste spezifiziert werden. Der Bediener kann anschließend eine Antwort geben, in dem er einer von mehreren vordefinierten Antwortmöglichkeiten auswählt. Die Antwort kann dann bspw. im Rahmen der Implementierung eines Zustands des Service über die interne Variable InteractAnswerID ausgewertet werden.

18.3.3.7 Vorbereitung bzw. freigeben verwendeter Sensoren/Aktoren

Ein Service wird in den allermeisten Fällen auf einen oder mehrere Sensoren und/oder Aktoren (bspw. Instanzen der FBs AnaView oder AnaDrv) zugreifen. Daher ist sicherzustellen, dass diese Sensoren und Aktoren in einem entsprechenden nutzbaren Zustand sind, bevor der Service vom Zustand Offline in Operator oder Automatic geschaltet wird. Hierzu definiert der Baustein die Methode regainActuatorControls, die im Rahmen eines entsprechenden Zustandsübergangs aufgerufen wird. Wird diese Methode überschrieben, können hier speziellen Algorithmen implementiert werden, um bspw. alle genutzten Sensoren/Aktoren ebenfalls in den Zustand Operator bzw. Automatic zu schalten.

Äquivalent steht die Methode releaseActuatorControls bereit, um genutzte Sensoren/Aktoren falls notwendig freizugeben, sobald ein Service deaktiviert und in den Zustand Offline überführt wird.

Hinweis:

Die beiden Methoden stehen bei ‚Rechtsklick -> Objekt hinzufügen -> Methode...‘ automatisch zur Auswahl.

18.3.4 Source Mode

18.3.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

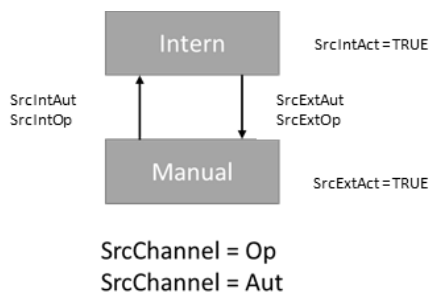


Fig. 137 Zustandsdiagramm Source Mode

18.3.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

18.3.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

18.3.5 Operation Mode

18.3.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

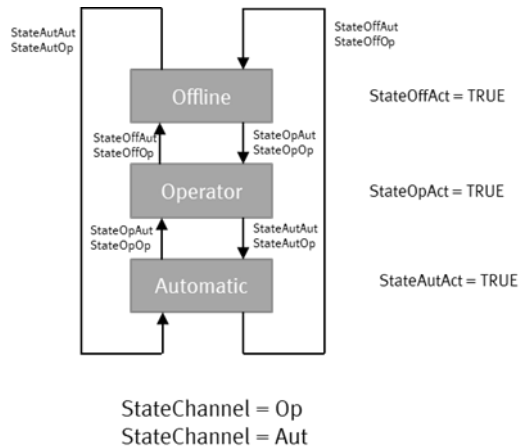


Fig. 138 Zustandsdiagramm Operation Mode

18.3.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

18.3.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.

- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

18.3.6 Symbol

18.3.6.1 Beschreibung

Element	Beschreibung	Regel
Tagname	Name der Einheit, wird in der Visualisierung angegeben.	Max. 16 Zeichen.
Operation Mode	 : Offline Mode ist aktiv  : Operator Mode ist aktiv  : Automatic Mode ist aktiv	Siehe Beschreibung Operation Mode.
Current State	Gibt den aktuellen Zustand der Statemaschine aus.	
Actual Step	Text, welcher in den einzelnen States implementiert werden kann, wird hier dargestellt.	Max. 24 Zeichen erlaubt.
License marker	Wird keine gültige Lizenz für das PA Toolkit gefunden und ist die Demozeit von 2 Std. abgelaufen wird der FB in einen sicheren Zustand gesetzt und im Symbol erscheint der Text „No license found!“.	Unsichtbar, bis Lizenz abgelaufen ist.

Tab. 118 Service Symbolbeschreibung

18.3.6.2 hServiceDefault

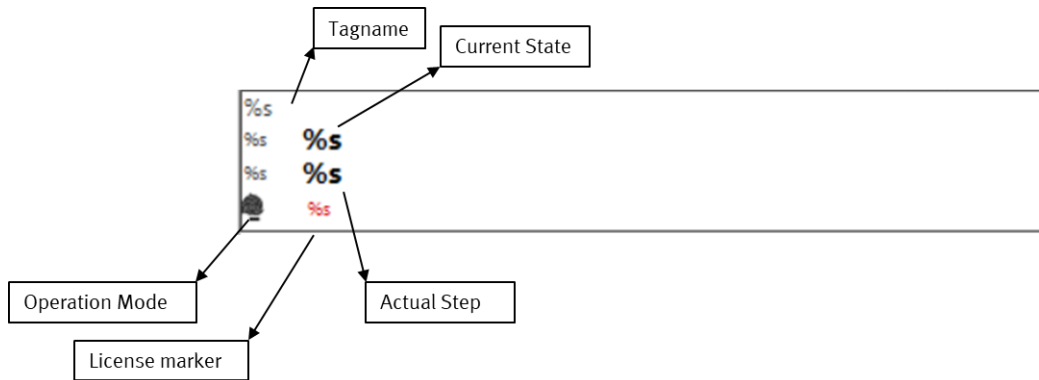


Fig. 139 Symbol hServiceDefault

18.3.7 Faceplates

18.3.7.1 Main

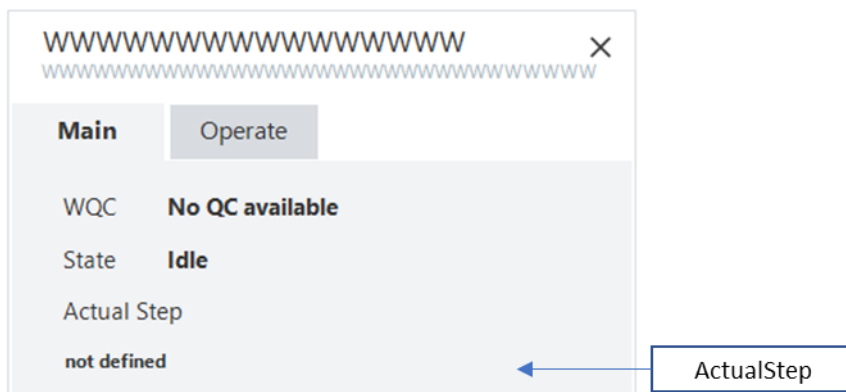


Fig. 140 Service Main Faceplate

18.3.7.2 Operate

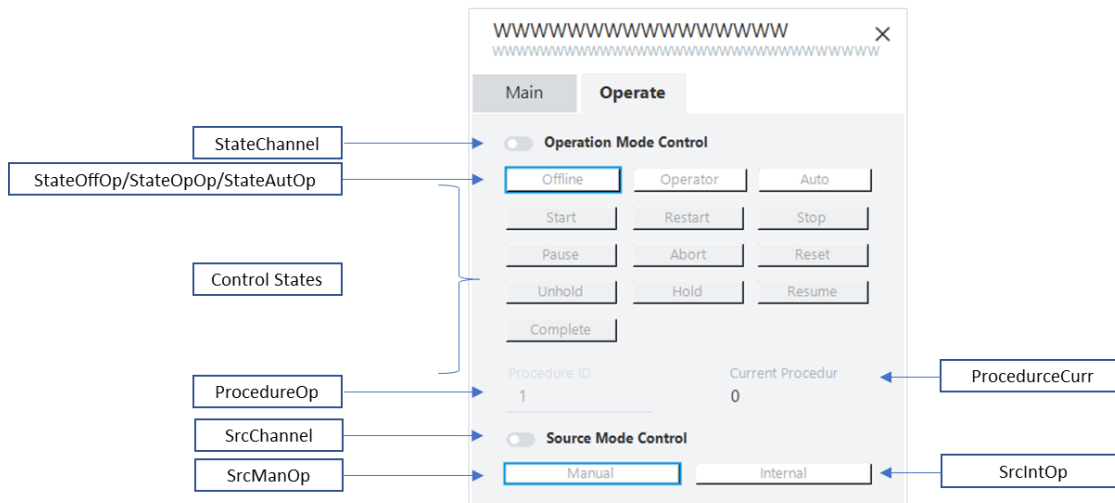


Fig. 141 Service Operation Faceplate

19 Serviceprozeduren

19.1 ServiceProcedure

19.1.1 Kurzbeschreibung

Eine Prozedur bzw. Fahrweise repräsentiert eine spezifische Ausprägung eines Dienstes. Bspw. kann ein Dienst Rühren über die zwei Prozeduren *kontinuierlich* und *zeitbeschränkt* verfügen. Der Funktionsbaustein (FB) ServiceProcedure kann zur Definition einer solchen Prozedur verwendet werden.

19.1.2 Grundfunktionen

19.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

19.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 119 Worst Quality Code Enumeration

19.1.3 Funktionen

19.1.3.1 Charakterisierung von Prozeduren

Eine Prozedur wird durch eine ID eindeutig (im Rahmen des Dienstes, für den die Prozedur definiert wird) beschrieben. Diese ID wird bspw. genutzt, wenn die Prozedur für einen Dienst angefordert wird (über einen der Eingänge ProcedureOp/Int/Ext des FBs ServiceControl). Zur Vorgabe dieser ID dient der Eingang ProcedureID.

Hinweis:

Zu beachten ist, dass für jeden Dienst zwingend eine Prozedur mit der ID 1 definiert werden muss, die als Default-Prozedur agiert.

Zusätzlich kann für eine Prozedur über den Eingang IsSelfCompleting angegeben werden, ob diese kontinuierlich oder selbstbeendend ist. Bei selbstbeendenden Prozeduren erfolgt der Übergang zwischen den Zuständen Execute und

Completing (s. Dokumentation zum FB ServiceControl) automatisch, während dieser Übergang bei kontinuierlichen Prozeduren nur durch ein entsprechendes Kommando (bspw. durch den Operator) ausgelöst werden kann.

19.1.3.2 Zuordnung von Dienst und Prozedur

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst und Prozedur. Um diese Zuordnung deutlich zu machen, ist eine Prozedur grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Zusätzlich ist dem Prozedur-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen. Folgender Quellcode-Ausschnitt zeigt exemplarisch Deklaration, Initialisierung und Aufruf von zwei Prozeduren.

(* Beispieldeklaration *)

```
FUNCTION_BLOCK Rühren EXTENDS ServiceControl  
VAR
```

```
    Kontinuierlich: ServiceProcedure := (TagName := 'Kontinuierlich', TagDescription := '', ProcedureID  
:= 1, IsSelfCompleting := FALSE);  
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '', ProcedureID := 2,  
IsSelfCompleting := TRUE);  
END_VAR
```

(* Beispielimplementierung ST *)

```
SUPER^();  
Kontinuierlich(Service := THIS^);  
Zeitbeschränkt(Service := THIS^);
```

19.1.3.3 Abfrage der aktiven Prozedur bei der Implementierung eines Dienstes

Ein Prozedur-Baustein zeigt über den Ausgang IsActive an, ob die durch ihn repräsentierte Prozedur gerade aktiv ist oder nicht. Dieser Ausgang kann im Rahmen der Implementierung eines Dienstes ausgewertet werden, um die eigentliche Implementierung der Logik für eine Prozedur vorzunehmen, wie folgendes Beispiel zeigt:

(* Beispielimplementierung des Zustands Execute eines Dienstes Rühren *)

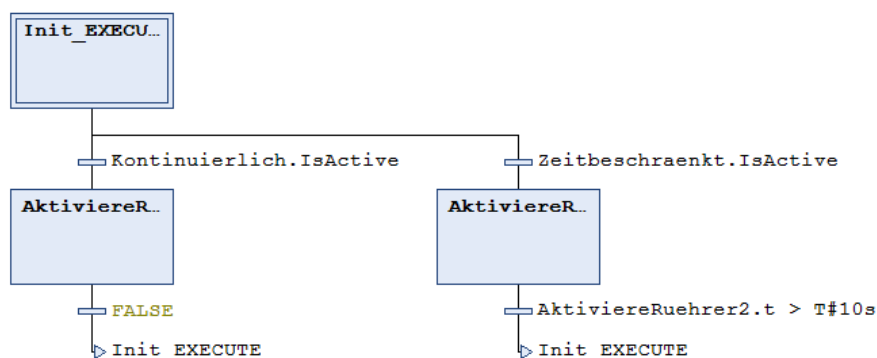


Fig. 142 Beispiel Zustand Execute

20 Analoge Dienstparameter

20.1 AnaConfigParam

20.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaConfigParam kann zur Definition eines analogen Konfigurationsparameters (Gleitkommazahl) für einen Dienst genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ REAL bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Konfigurationsparameter vor allem zur Vorgabe von Werten, die für die grundlegende Konfiguration eines Dienstes notwendig sind und sich während der Nutzung eines Dienstes nicht ändern dürfen (bspw. die Auswahl eines für einen Dosieren-Dienst zu nutzenden Ventils).

20.1.2 Grundfunktionen

20.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

20.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 120 Worst Quality Code Enumeration

20.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

20.1.3 Funktionen

20.1.3.1 Zuordnung von Dienst und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

```
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
```

```
VAR
```

```
    Parameter1: AnaConfigParam := (TagName := 'Parameter1', TagDescription := '');
```

```
END_VAR
```

(* Beispielimplementierung ST *)

```
SUPER^();
```

```
Parameter1(Service := THIS^);
```

20.1.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden. Die eingestellten Skalierungswerte (VMin und VMax) sind für alle Eingänge gültig, ebenfalls VUnit.

20.1.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert wird mit Hilfe der Normierungswerte („VMin“ und „VMax“) in den Ausgangswert „VOut“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte stehen als Eingänge am FB zur Verfügung. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant.

20.1.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

20.1.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

20.1.3.6 Einheit für „VOp“, „VInt“, „VExt“ und „VOut“

Für die Ein- und Ausgangswerte kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

20.1.3.7 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

20.1.3.8 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScImin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

20.1.4 Source Mode

20.1.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

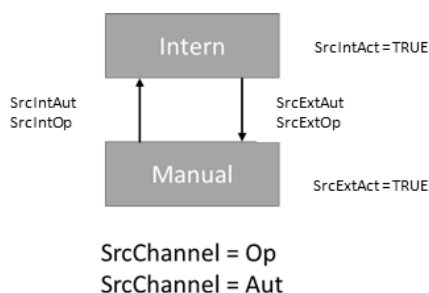


Fig. 143 Zustandsdiagramm Source Mode

20.1.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

20.1.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

20.1.5 Operation Mode

20.1.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

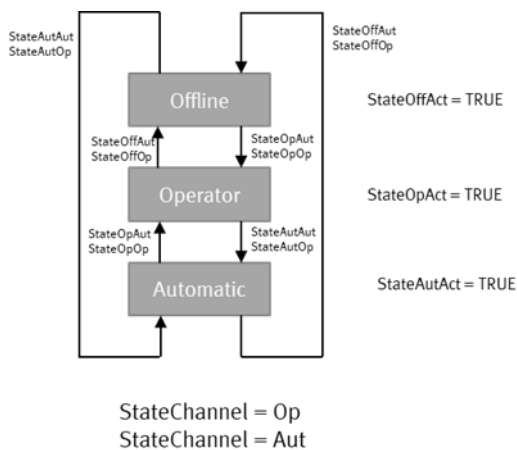


Fig. 144 Zustandsdiagramm Operation Mode

20.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

20.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

20.1.5.4 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter eineindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

20.2 AnaProcedureParam

20.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaProcedureParam kann zur Definition eines analogen Prozedurparameters (Gleitkommazahl) für eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ REAL bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Prozedurparameter vor allem zur Vorgabe von Werten, die für eine bestimmte Prozedur eines Dienstes notwendig sind und sich während der Ausführung der Prozedur nicht ändern dürfen (bspw. die Vorgabe einer Zeit für eine zeitbeschränkte Dosieren-Prozedur).

20.2.2 Grundfunktionen

20.2.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

20.2.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 121 Worst Quality Code Enumeration

20.2.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

20.2.3 Funktionen

20.2.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

```
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
```

```
VAR
```

```
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);
```

```
    Zeit: AnaProcedureParam := (TagName := 'Zeit', TagDescription := '', ProcedureID := 1);  
END_VAR
```

(* Beispielimplementierung ST *)

```
SUPER^();
```

```
Zeitbeschränkt(Service := THIS^);
```

```
Zeit(Service := THIS^);
```

20.2.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden. Die eingestellten Skalierungswerte (VMin und VMax) sind für alle Eingänge gültig, ebenfalls VUnit.

20.2.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert wird mit Hilfe der Normierungswerte („VMin“ und „VMax“) in den Ausgangswert „VOut“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte stehen als Eingänge am FB zur Verfügung. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant.

20.2.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

20.2.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

20.2.3.6 Einheit für „VOp“, „VInt“, „VExt“ und „VOut“

Für die Ein- und Ausgangswerte kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

20.2.3.7 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

20.2.3.8 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScMin und VScMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

20.2.4 Source Mode

20.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

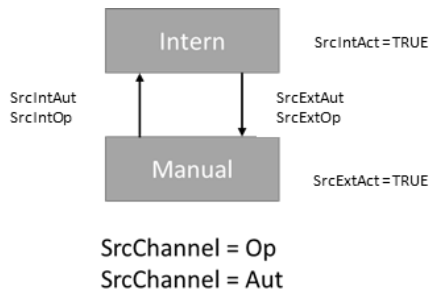


Fig. 145 Zustandsdiagramm Source Mode

20.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

20.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

20.2.5 Operation Mode

20.2.5.1 Kurzbeschreibung

Der Operation Mode ist einen Zustandsautomaten mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

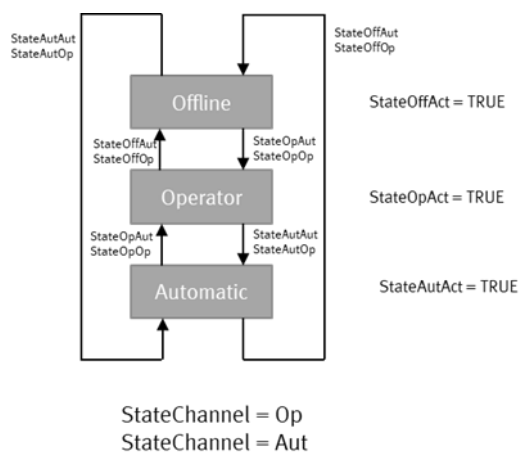


Fig. 146 Zustandsdiagramm Operation Mode

20.2.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.

- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

20.2.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

20.2.5.4 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter uneindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

20.3 AnaProcessValueIn

20.3.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaProcessValueIn kann zum kontinuierlichen Übertragen eines externen, analogen Wertes (Gleitkommazahl) an eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an.

20.3.2 Grundfunktionen

20.3.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'TagName'.

20.3.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle WQCs aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 122 Worst Quality Code Enumeration

20.3.3 Funktionen

20.3.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

(* Beispieldeklaration *)

```
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
```

```
VAR
```

```
    Zeitbeschaenkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);
```

```
    Fuellstand: AnaProcessValueIn := (TagName := 'Füllstand', TagDescription := '', ProcedureID := 1);
```

```
END_VAR
```



```
(* Beispielimplementierung ST *)  
SUPER^();  
Zeitbeschränkt(Service := THIS^);  
Fuellstand();
```

20.3.3.2 Eingangswert „VExt“

Der Eingangswert wird durch die Variable „VExt“ repräsentiert und wird als OPC UA Variable zur Verfügung gestellt.

20.3.3.3 Ausgangswert „VOut“

Der Eingangswert wird ohne Umrechnung auf den Ausgangswert „VOut“ gegeben und als Ausgang am FB ausgegeben.

20.3.3.4 Einheit für „VExt“ und „VOut“

Für den Ein- und Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

20.3.3.5 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScIMin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

20.4 AnaProcessValueOut

20.4.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaProcessValueOut dient zum kontinuierlichen Rückmelden eines analogen Wertes (Gleitkommazahl) aus einer Dienst-Prozedur.

20.4.2 Grundfunktionen

20.4.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

20.4.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 123 Worst Quality Code Enumeration

20.4.3 Funktionen

20.4.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Rühren EXTENDS ServiceControl  
VAR  
    Zeitbeschaenkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);  
    Fuellstand: AnaProcessValueOut := (TagName := 'Fuellstand', TagDescription := '', ProcedureID :=  
1);  
END_VAR  
  
(* Beispielimplementierung ST *)  
SUPER^();  
Zeitbeschränkt(Service := THIS^);  
Fuellstand();
```

20.4.3.2 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer analogen Eingangskarte verbunden werden. Über die Normierungswerte „FVNormMin“ und „FVNormMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind ebenfalls Eingänge am FB.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert. Das bedeutet, dass der simulierte Wert nicht skaliert wird. Es wird nur überprüft, ob der simulierte Werte die Grenzwerte von „VNormMin“ und „VNormMax“ nicht unterschreitet bzw. überschreitet. Werden die Grenzwerte nicht eingehalten wird der entsprechende Grenzwert als „V“ ausgegeben.

20.4.3.3 Ausgangswert „V“

Der Eingangswert „FV“ wird mit Hilfe der Normierungswerte („FVNormMin“, „FVNormMax“, „VNormMin“ und „VNormMax“) in den Ausgangswert „V“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswert als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „V“ seine Normierungswerte („VNormMin“ und „VNormMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

Wird die Simulation verwendet wird als WQC „Simulated Value“ eingestellt. Der simulierte Wert wird nicht skaliert am Ausgang „V“ ausgegeben. D.h. es findet nur eine Grenzwertüberwachung und keine Umrechnung statt.

20.4.3.4 Umrechnung Eingangswert „FV“ in Ausgangswert „V“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswerte benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Eingang: FVNormMin und FVNormMax

Ausgang: VNormMin und VNormMax

20.4.3.5 Einheit für „V“

Für den Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

20.4.3.6 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von FVNormMax und FVNormMin nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

20.4.3.7 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScImin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Ist der Eingang FullScIEn auf TRUE gesetzt so werden automatisch die Normierungswerte VNormMin und VNormMax als Anzeigegrenzen verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Normierungswerte gesetzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

20.4.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code

Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin oder VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax - VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 124 AnaProcessValueOut spezifischer Worst Quality Code

20.4.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 125 AnaProcessValueOut persistente Variablen

20.5 AnaReportValue

20.5.1 Kurzbeschreibung

Der Funktionsbaustein (FB) AnaReportValue dient zur Bereitstellung eines analogen Wertes (Gleitkommazahl) aus einer Dienst-Prozedur zu Dokumentations- bzw. Nachweiszwecken. Dieser FB erweitert die Funktionalität eines AnaProcessValueOut derart, dass der Wert des Parameters („V“) eingefroren wird, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Dies ermöglicht es einem angeschlossenen Dokumentations-System, den Wert des Parameters in diesen Zuständen bspw. für Nachweiszwecke auszulesen.

20.5.2 Grundfunktionen

20.5.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

20.5.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 126 Worst Quality Code Enumeration

20.5.3 Funktionen

20.5.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

```
(* Beispieldeklaration *)
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
VAR
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
    ProcedureID := 1, IsSelfCompleting := TRUE);
    Fuellstand: AnaReportValue := (TagName := 'Fuellstand', TagDescription := '', ProcedureID := 1);
END_VAR

(* Beispielimplementierung ST *)
SUPER^();
Zeitbeschränkt(Service := THIS^);
Fuellstand(Service := THIS^);
```

20.5.3.2 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer analogen Eingangskarte verbunden werden. Über die Normierungswerte „FVNormMin“ und „FVNormMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind ebenfalls Eingänge am FB.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert. Das bedeutet, dass der simulierte Wert nicht skaliert wird. Es wird nur überprüft, ob der simulierte Werte die Grenzwerte von „VNormMin“ und „VNormMax“ nicht unterschreitet bzw. überschreitet. Werden die Grenzwerte nicht eingehalten wird der entsprechende Grenzwert als „V“ ausgegeben.

20.5.3.3 Ausgangswert „V“

Der Eingangswert „FV“ wird mit Hilfe der Normierungswerte („FVNormMin“, „FVNormMax“, „VNormMin“ und „VNormMax“) in den Ausgangswert „V“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswert als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „V“ seine Normierungswerte („VNormMin“ und „VNormMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

Wird die Simulation verwendet wird als WQC „Simulated Value“ eingestellt. Der simulierte Wert wird nicht skaliert am Ausgang „V“ ausgegeben. D.h. es findet nur eine Grenzwertüberwachung und keine Umrechnung statt.

Der Ausgangswert („V“) wird eingefroren, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Befindet sich der Dienst im Zustand Offline, wird der Wert auf 0 gesetzt.

20.5.3.4 Umrechnung Eingangswert „FV“ in Ausgangswert „V“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswerte benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Eingang: FVNormMin und FVNormMax

Ausgang: VNormMin und VNormMax

20.5.3.5 Einheit für „V“

Für den Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

20.5.3.6 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von FVNormMax und FVNormMin nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

20.5.3.7 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScIMin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Ist der Eingang FullScIEn auf TRUE gesetzt so werden automatisch die Normierungswerte VNormMin und VNormMax als Anzeigegrenzen verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Normierungswerte gesetzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

20.5.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax - VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 127 AnaReportValue spezifischer Worst Quality Code

20.5.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 128 AnaReportValue persistente Variablen

21 Binäre Dienstparameter

21.1 BinConfigParam

21.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinConfigParam kann zur Definition eines binären (booleschen) Konfigurationsparameters für einen Dienst genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ BOOL bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Konfigurationsparameter vor allem zur Vorgabe von Werten, die für die grundlegende Konfiguration eines Dienstes notwendig sind und sich während der Nutzung eines Dienstes nicht ändern dürfen (bspw. die Auswahl eines für einen Dosieren-Dienst zu nutzenden Ventils).

21.1.2 Grundfunktionen

21.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

21.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 129 Worst Quality Code Enumeration

21.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

21.1.3 Funktionen

21.1.3.1 Zuordnung von Dienst und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

FUNCTION_BLOCK Rühren EXTENDS ServiceControl

VAR

Parameter1: BinConfigParam := (TagName := 'Parameter1', TagDescription := '');

END_VAR

(* Beispielimplementierung ST *)

SUPER^();

Parameter1(Service := THIS^);

21.1.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden.

21.1.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert als „VOut“ als Ausgang am FB ausgegeben. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant.

21.1.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

21.1.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

21.1.4 Source Mode

21.1.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

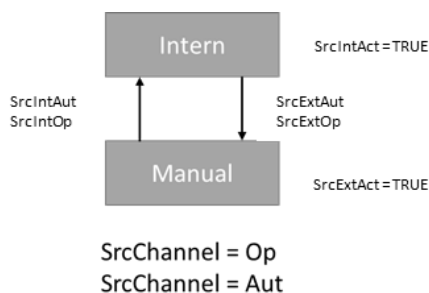


Fig. 147 Zustandsdiagramm Source Mode

21.1.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

21.1.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

21.1.5 Operation Mode

21.1.5.1 Kurzbeschreibung

Der Operation Mode ist einen Zustandsautomaten mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

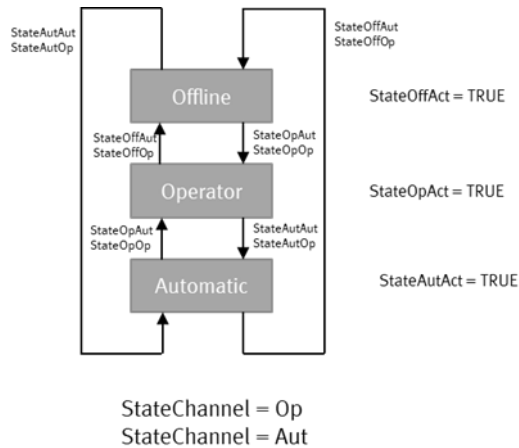


Fig. 148 Zustandsdiagramm Operation Mode

21.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

21.1.5.3 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter eindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

21.2 BinProcedureParam

21.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinProcedureParam kann zur Definition eines binären (booleschen) Prozedurparameters für eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ BOOL bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Prozedurparameter vor allem zur Vorgabe von Werten, die für eine bestimmte Prozedur eines Dienstes notwendig sind und sich während der Ausführung der Prozedur nicht ändern dürfen (bspw. die Vorgabe einer Zeit für eine zeitbeschränkte Dosieren-Prozedur).

21.2.2 Grundfunktionen

21.2.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

21.2.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 130 Worst Quality Code Enumeration

21.2.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden

Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert. Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

21.2.3 Funktionen

21.2.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

```
(* Beispieldeklaration *)
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
VAR
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
    ProcedureID := 1, IsSelfCompleting := TRUE);
    Zeit: BinProcedureParam := (TagName := 'Zeit', TagDescription := '', ProcedureID := 1);
END_VAR

(* Beispielimplementierung ST *)
SUPER^();
Zeitbeschränkt(Service := THIS^);
Zeit(Service := THIS^);
```

21.2.3.2 Eingangswert (Operator /Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden.

21.2.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert als „VOut“ als Ausgang am FB ausgegeben. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant.

21.2.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

21.2.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

21.2.4 Source Mode

21.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

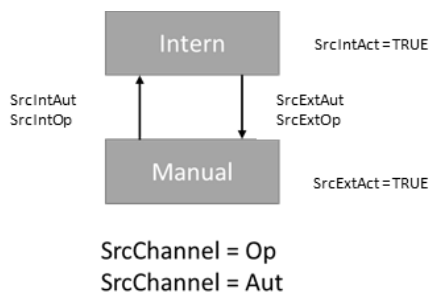


Fig. 149 Zustandsdiagramm Source Mode

21.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

21.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

21.2.5 Operation Mode

21.2.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

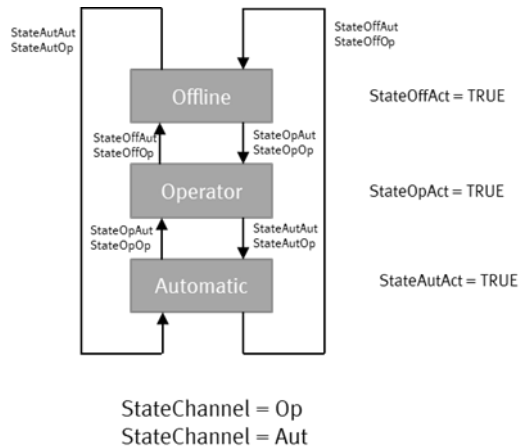


Fig. 150 Zustandsdiagramm Operation Mode

21.2.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

21.2.5.3 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter eindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

21.3 BinProcessValueIn

21.3.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinProcessValueIn kann zum kontinuierlichen Übertragen eines externen, binären (booleschen) Wertes an eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an.

21.3.2 Grundfunktionen

21.3.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

21.3.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 131 Worst Quality Code Enumeration

21.3.3 Funktionen

21.3.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Rühren EXTENDS ServiceControl  
VAR  
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);  
    Fuellstand: BinProcessValueIn := (TagName := 'Füllstand', TagDescription := '', ProcedureID := 1);  
END_VAR  
  
(* Beispielimplementierung ST *)  
SUPER^();  
Zeitbeschränkt(Service := THIS^);  
Fuellstand();
```

21.3.3.2 Eingangswert „VExt“

Der Eingangswert wird durch die Variable „VExt“ repräsentiert und wird als OPC UA Variable zur Verfügung gestellt.

21.3.3.3 Ausgangswert „VOut“

Der Eingangswert wird auf den Ausgangswert „VOut“ gegeben und als Ausgang am FB ausgegeben.

21.3.3.4 Statustext „VState0“ und „VState1“

Am FB stehen die Eingänge „VState0“ und „VState1“ als zwei String-Variable zur Verfügung. An diesen Eingängen können Texte für die beiden Zustände hinterlegt werden. Diese beschreiben zusätzlich die Zustände in Worten. So kann z.B. für den Wert TRUE der Eingang VState1 mit dem Text „Lampe an“ belegt werden.

Für jeden Statustext stehen 24 Zeichen zur Verfügung.

21.4 BinProcessValueOut

21.4.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinProcessValueOut dient zum kontinuierlichen Rückmelden eines binären (booleschen) Wertes aus einer Dienst-Prozedur.

21.4.2 Grundfunktionen

21.4.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

21.4.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 132 Worst Quality Code Enumeration

21.4.3 Funktionen

21.4.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

```
(* Beispieldeklaration *)
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
VAR
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
    ProcedureID := 1, IsSelfCompleting := TRUE);
    Fuellstand: BinProcessValueOut := (TagName := 'Fuellstand', TagDescription := '', ProcedureID := 1);
END_VAR

(* Beispielimplementierung ST *)
SUPER^();
Zeitbeschränkt(Service := THIS^);
Fuellstand();
```

21.4.3.2 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer digitalen Eingangskarte verbunden werden. Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert

21.4.3.3 Statustext „VState0“ und „VState1“

Am FB stehen die Eingänge „State0“ und „State1“ als zwei String-Variable zur Verfügung. An diesen Eingängen können Texte für die beiden Zustände hinterlegt werden. Diese beschreiben zusätzlich die Zustände in Worten. So kann z.B. für den Wert TRUE der Eingang VState1 mit dem Text „Tank ist voll“ belegt werden. Für jeden Statustext stehen 24 Zeichen zur Verfügung.

21.4.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 133 BinProcessValueOut spezifischer Worst Quality Code

21.5 BinReportValue

21.5.1 Kurzbeschreibung

Der Funktionsbaustein (FB) BinReportValue dient zum Bereitstellen eines binären (booleschen) Wertes aus einer Dienst-Prozedur zu Dokumentations- bzw. Nachweiszwecken. Dieser FB erweitert die Funktionalität eines BinProcessValueOut derart, dass der Wert des Parameters („V“) eingefroren wird, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Dies ermöglicht es einem angeschlossenen Dokumentations-System, den Wert des Parameters in diesen Zuständen bspw. für Nachweiszwecke auszulesen.

21.5.2 Grundfunktionen

21.5.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

21.5.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 134 Worst Quality Code Enumeration

21.5.3 Funktionen

21.5.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

```
(* Beispieldeklaration *)
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
VAR
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
    ProcedureID := 1, IsSelfCompleting := TRUE);
    Fuellstand: BinReportValue := (TagName := 'Fuellstand', TagDescription := '', ProcedureID := 1);
END_VAR

(* Beispielimplementierung ST *)
SUPER^();
Zeitbeschränkt(Service := THIS^);
Fuellstand(Service := THIS^);
```

21.5.3.2 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer digitalen Eingangskarte verbunden werden. Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert.

21.5.3.3 Statustext „VState0“ und „VState1“

Am FB stehen die Eingänge „State0“ und „State1“ als zwei String-Variable zur Verfügung. An diesen Eingängen können Texte für die beiden Zustände hinterlegt werden. Diese beschreiben zusätzlich die Zustände in Worten. So kann z.B. für den Wert TRUE der Eingang VState1 mit dem Text „Tank ist voll“ belegt werden. Für jeden Statustext stehen 24 Zeichen zur Verfügung.

21.5.3.4 Ausgangswert „V“

Der Ausgangswert („V“) wird eingefroren, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Befindet sich der Dienst im Zustand Offline, wird der Wert auf FALSE gesetzt.

21.5.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 135 BinReportValue spezifischer Worst Quality Code

22 Digitale Dienstparameter

22.1 DIntConfigParam

22.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntConfigParam kann zur Definition eines analogen Konfigurationsparameters (Double-Integer) für einen Dienst genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ DINT bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Konfigurationsparameter vor allem zur Vorgabe von Werten, die für die grundlegende Konfiguration eines Dienstes notwendig sind und sich während der Nutzung eines Dienstes nicht ändern dürfen (bspw. die Auswahl eines für einen Dosieren-Dienst zu nutzenden Ventils).

22.1.2 Grundfunktionen

22.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

22.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 136 Worst Quality Code Enumeration

22.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

22.1.3 Funktionen

22.1.3.1 Zuordnung von Dienst und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

FUNCTION_BLOCK Rühren EXTENDS ServiceControl

VAR

Parameter1: DIntConfigParam := (TagName := 'Parameter1', TagDescription := '');

END_VAR

(* Beispielimplementierung ST *)

SUPER^();

Parameter1(Service := THIS^);

22.1.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden. Die eingestellten Skalierungswerte (VMin und VMax) sind für alle Eingänge gültig, ebenfalls VUnit.

22.1.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert wird mit Hilfe der Normierungswerte („VMin“ und „VMax“) in den Ausgangswert „VOut“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte stehen als Eingänge am FB zur Verfügung. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant.

22.1.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

22.1.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

22.1.3.6 Einheit für „VOp“, „VInt“, „VExt“ und „VOut“

Für die Ein- und Ausgangswerte kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

22.1.3.7 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

22.1.3.8 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScImin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

22.1.4 Source Mode

22.1.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

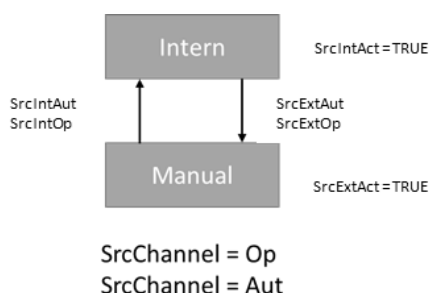


Fig. 151 Zustandsdiagramm Source Mode

22.1.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

22.1.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

22.1.5 Operation Mode

22.1.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

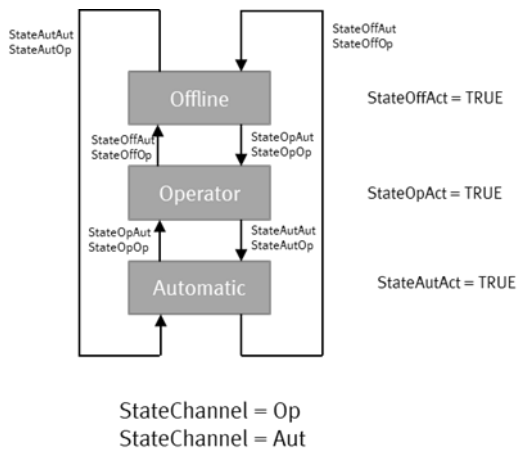


Fig. 152 Zustandsdiagramm Operation Mode

22.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

22.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

22.1.5.4 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter uneindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

22.2 DIntProcedureParam

22.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntProcedureParam kann zur Definition eines analogen Prozedurparameters (Double-Integer) für eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ DINT bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Prozedurparameter vor allem zur Vorgabe von Werten, die für eine bestimmte Prozedur eines Dienstes notwendig sind und sich während der Ausführung der Prozedur nicht ändern dürfen (bspw. die Vorgabe einer Zeit für eine zeitbeschränkte Dosieren-Prozedur).

22.2.2 Grundfunktionen

22.2.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

22.2.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 137 Worst Quality Code Enumeration

22.2.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

22.2.3 Funktionen

22.2.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

FUNCTION_BLOCK Rühren EXTENDS ServiceControl

VAR

Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
ProcedureID := 1, IsSelfCompleting := TRUE);

Zeit: DIntProcedureParam := (TagName := 'Zeit', TagDescription := '', ProcedureID := 1);
END_VAR

(* Beispielimplementierung ST *)

SUPER^();

Zeitbeschränkt(Service := THIS^);

Zeit(Service := THIS^);

22.2.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden. Die eingestellten Skalierungswerte (VMin und VMax) sind für alle Eingänge gültig, ebenfalls VUnit.

22.2.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert wird mit Hilfe der Normierungswerte („VMin“ und „VMax“) in den Ausgangswert „VOut“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte stehen als Eingänge am FB zur Verfügung. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant.

22.2.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

22.2.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

22.2.3.6 Einheit für „VOp“, „VInt“, „VExt“ und „VOut“

Für die Ein- und Ausgangswerte kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

22.2.3.7 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

22.2.3.8 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScMin und VScMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

22.2.4 Source Mode

22.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

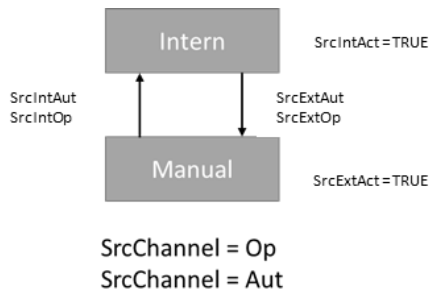


Fig. 153 Zustandsdiagramm Source Mode

22.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

22.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

22.2.5 Operation Mode

22.2.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

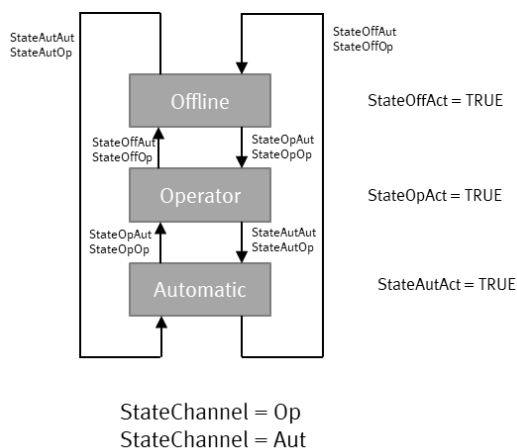


Fig. 154 Zustandsdiagramm Operation Mode

22.2.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/“StateOpAut“/“StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

22.2.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/“StateOpOp“/“StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

22.2.5.4 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter eineindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

22.3 DIntProcessValueIn

22.3.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntProcessValueIn kann zum kontinuierlichen Übertragen eines externen, analogen Wertes (Double-Integer) an eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an.

22.3.2 Grundfunktionen

22.3.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

22.3.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 138 Worst Quality Code Enumeration

22.3.3 Funktionen

22.3.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Rühren EXTENDS ServiceControl  
VAR  
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);  
    Fuellstand: DintProcessValueIn := (TagName := 'Füllstand', TagDescription := '', ProcedureID := 1);  
END_VAR  
  
(* Beispielimplementierung ST *)  
SUPER^();  
Zeitbeschränkt(Service := THIS^);  
Fuellstand();
```

22.3.3.2 Eingangswert „VExt“

Der Eingangswert wird durch die Variable „VExt“ repräsentiert und wird als OPC UA Variable zur Verfügung gestellt.

22.3.3.3 Ausgangswert „VOut“

Der Eingangswert wird ohne Umrechnung auf den Ausgangswert „VOut“ gegeben und als Ausgang am FB ausgegeben.

22.3.3.4 Einheit für „VExt“ und „VOut“

Für den Ein- und Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

22.3.3.5 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScImin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet.

22.4 DIntProcessValueOut

22.4.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntProcessValueOut dient zum kontinuierlichen Rückmelden eines analogen Wertes (Double-Integer) aus einer Dienst-Prozedur.

22.4.2 Grundfunktionen

22.4.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

22.4.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 139 Worst Quality Code Enumeration

22.4.3 Funktionen

22.4.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

```
(* Beispieldeklaration *)
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
VAR
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
    ProcedureID := 1, IsSelfCompleting := TRUE);
    Fuellstand: DIntProcessValueOut := (TagName := 'Fuellstand', TagDescription := '', ProcedureID :=
1);
END_VAR

(* Beispielimplementierung ST *)
SUPER^();
Zeitbeschränkt(Service := THIS^);
Fuellstand();
```

22.4.3.2 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer analogen Eingangskarte verbunden werden. Über die Normierungswerte „FVNormMin“ und „FVNormMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind ebenfalls Eingänge am FB.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert. Das bedeutet, dass der simulierte Wert nicht skaliert wird. Es wird nur überprüft, ob der simulierte Werte die Grenzwerte von „VNormMin“ und „VNormMax“ nicht unterschreitet bzw. überschreitet. Werden die Grenzwerte nicht eingehalten wird der entsprechende Grenzwert als „V“ ausgegeben.

22.4.3.3 Ausgangswert „V“

Der Eingangswert „FV“ wird mit Hilfe der Normierungswerte („FVNormMin“, „FVNormMax“, „VNormMin“ und „VNormMax“) in den Ausgangswert „V“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswert als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „V“ seine Normierungswerte („VNormMin“ und „VNormMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

Wird die Simulation verwendet wird als WQC „Simulated Value“ eingestellt. Der simulierte Wert wird nicht skaliert am Ausgang „V“ ausgegeben. D.h. es findet nur eine Grenzwertüberwachung und keine Umrechnung statt.

22.4.3.4 Umrechnung Eingangswert „FV“ in Ausgangswert „V“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswerte benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Eingang: FVNormMin und FVNormMax

Ausgang: VNormMin und VNormMax

22.4.3.5 Einheit für „V“

Für den Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

22.4.3.6 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von FVNormMax und FVNormMin nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

22.4.3.7 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScIMin und VScIMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Ist der Eingang FullScIEn auf TRUE gesetzt so werden automatisch die Normierungswerte VNormMin und VNormMax als Anzeigegrenzen verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Normierungswerte gesetzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

22.4.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSImEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax - VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 140 DIntProcessValueOut spezifischer Worst Quality Code

22.4.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> **Add Object** -> **Persistent Variables** ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 141 DIntProcessValueOut

22.5 DIntReportValue

22.5.1 Kurzbeschreibung

Der Funktionsbaustein (FB) DIntReportValue dient zum Bereitstellen eines analogen Wertes (Double-Integer) aus einer Dienst-Prozedur zu Dokumentations- bzw. Nachweiszwecken. Dieser FB erweitert die Funktionalität eines DIntProcessValueOut derart, dass der Wert des Parameters („V“) eingefroren wird, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Dies ermöglicht es einem angeschlossenen Dokumentations-System, den Wert des Parameters in diesen Zuständen bspw. für Nachweiszwecke auszulesen.

22.5.2 Grundfunktionen

22.5.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

22.5.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 142 Worst Quality Code Enumeration

22.5.3 Funktionen

22.5.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

```
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
```

```
VAR
```

```
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);
```

```
    Fuellstand: DIntReportValue := (TagName := 'Fuellstand', TagDescription := '', ProcedureID := 1);  
END_VAR
```

(* Beispielimplementierung ST *)

```
SUPER^();
```

```
Zeitbeschränkt(Service := THIS^);
```

```
Fuellstand(Service := THIS^);
```

22.5.3.2 Eingangswert „FV“ und „VSim“ (Fieldvalue oder simulierter Wert)

Der Eingangswert wird durch die Variable „FV“ repräsentiert und als Eingang am FB zur Verfügung gestellt. Dieser Eingang kann mit einer Variablen oder einer Adresse einer analogen Eingangskarte verbunden werden. Über die Normierungswerte „FVNormMin“ und „FVNormMax“ wird die Skalierung des Eingangswertes eingestellt. Diese beiden Werte sind ebenfalls Eingänge am FB.

Ist kein Eingangswert verfügbar, kann über den Eingang „VSim“ am FB ein Wert simuliert werden. Die Simulation wird über das Eingangsbit „VSimEn“ (=TRUE) aktiviert. Hier wird direkt der Ausgangswert „V“ simuliert. Das bedeutet, dass der simulierte Wert nicht skaliert wird. Es wird nur überprüft, ob der simulierte Werte die Grenzwerte von „VNormMin“ und „VNormMax“ nicht unterschreitet bzw. überschreitet. Werden die Grenzwerte nicht eingehalten wird der entsprechende Grenzwert als „V“ ausgegeben.

22.5.3.3 Ausgangswert „V“

Der Eingangswert „FV“ wird mit Hilfe der Normierungswerte („FVNormMin“, „FVNormMax“, „VNormMin“ und „VNormMax“) in den Ausgangswert „V“ umgerechnet und als Ausgang am FB ausgegeben. Die Normierungswerte von „V“ stehen ebenfalls wie die Normierungswerte des Eingangswert als Eingänge am FB zur Verfügung. Überschreitet bzw. unterschreitet „V“ seine Normierungswerte („VNormMin“ und „VNormMax“) so wird der entsprechende Normierungswert als Ausgang ausgegeben.

Wird die Simulation verwendet wird als WQC „Simulated Value“ eingestellt. Der simulierte Wert wird nicht skaliert am Ausgang „V“ ausgegeben. D.h. es findet nur eine Grenzwertüberwachung und keine Umrechnung statt.

Der Ausgangswert („V“) wird eingefroren, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Befindet sich der Dienst im Zustand Offline, wird der Wert auf 0 gesetzt.

22.5.3.4 Umrechnung Eingangswert „FV“ in Ausgangswert „V“

Um den Eingangswert in einen Ausgangswert umzurechnen, werden für Eingang sowie für den Ausgang jeweils die Normierungswerte benötigt, welche am FB als Eingänge eingegeben werden können.

Die Formel zur Umrechnung sieht wie folgt aus:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Eingang: FVNormMin und FVNormMax

Ausgang: VNormMin und VNormMax

22.5.3.5 Einheit für „V“

Für den Ausgangswert kann über die Eingangsvariable „VUnit“ am FB eine Einheit angegeben werden. Um eine Einheit anzugeben, wird die Enumeration „UnitSettings“ verwendet. Die Unit wird nur zur Darstellung verwendet und hat keinen Einfluss auf die Umrechnung.

22.5.3.6 Überwachung der Normierungswerte

Wie oben beschrieben begrenzen die Normierungswerte den Eingangswert und Ausgangswert und werden bei der Umrechnung herangezogen. Zusätzlich muss darauf geachtet werden, dass die minimalen Normierungswerte nicht größer sind als die dazugehörigen maximalen Normierungswerte. Andernfalls wird nicht garantiert, dass die Umrechnung und die Grafiken in der Visualisierung sinnvolle Werte enthalten.

Ebenfalls darf die Differenz von FVNormMax und FVNormMin nicht null ergeben, da nicht durch null geteilt werden kann. In diesem Fall wird am Ausgangswert „V“ null ausgegeben.

22.5.3.7 Skalierungswerte für die Visualisierung

Zum Festlegen der Anzeigegrenzen einer visuellen Darstellung des Ausgangswertes, werden jeweils eine Min und eine Max Variable (VScMin und VScMax) verwendet. Diese Variablen werden ausschließlich für die Darstellung graphischer Elemente in der Visualisierung verwendet. Ist der Eingang FullScEn auf TRUE gesetzt so werden automatisch die Normierungswerte VNormMin und VNormMax als Anzeigegrenzen verwendet. Werden die Anzeigegrenzen falsch gesetzt werden sie automatisch auf die Normierungswerte gesetzt. Siehe dazu Abschnitt FB-spezifischer Worst Quality Code.

22.5.4 FB-spezifischer Worst Quality Code (WQC)

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin oder VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax - VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 143 DIntReportValue spezifischer Worst Quality Code

22.5.5 Persistente Variablen

Persistenten Variablen werden in der Steuerung dauerhaft abgespeichert, so dass sie nach einem:

- Kaltstart
- Download des Programms
- Online-Change

trotzdem ihre aktuellen Werte nicht verlieren. Nur bei einem kompletten Reset (Ursprung) der Steuerung bekommen die Variablen ihre Initialwerte wieder.

Damit die Steuerung die Variablen persistent speichert, wird in CoDeSys über

Applikation -> Add Object -> Persistent Variables ein zusätzliches Objekt in den Strukturbaum eingefügt. In diesem Objekt können alle Instanzpfade der persistenten Variablen automatisch eingefügt werden.

Dazu muss das Objekt „persistente Variables“ geöffnet werden. In der oberen Menüleiste erscheint der Reiter **Declaration**. Über diesen kann **Add all Instancepaths** ausgewählt werden. Dadurch werden die entsprechenden Verknüpfungen erstellt. Kommt es zu Fehler, so muss der Namespace „FestoPAToolkit“ manuell hinzugefügt werden.

Hinweis:

- Es kann selbst eine Auswahl getroffen werden, welche Variablen persistent gespeichert werden soll.
- Es muss die Speicherkapazität für persistente Variablen jeder Steuerung berücksichtigt werden. Diese wird in den Handbüchern der jeweiligen Steuerung beschrieben.

Die folgende Tabelle listet die persistenten Variablen für diesen FB aus. Um einen problemlosen Ablauf zu ermöglichen, sollten diese auch als persistente Variablen in der Applikation verwendet werden.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 144 DIntReportValue persistente Variablen

23 Zeichenketten-Dienstparameter

23.1 StringConfigParam

23.1.1 Kurzbeschreibung

Der Funktionsbaustein (FB) StringConfigParam kann zur Definition eines String-basierten Konfigurationsparameters (Zeichenkette) für einen Dienst genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ STRING bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Konfigurationsparameter vor allem zur Vorgabe von Werten, die für die grundlegende Konfiguration eines Dienstes notwendig sind und sich während der Nutzung eines Dienstes nicht ändern dürfen (bspw. die Auswahl eines für einen Dosieren-Dienst zu nutzenden Ventils).

23.1.2 Grundfunktionen

23.1.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

23.1.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 145 Worst Quality Code Enumeration

23.1.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienungsstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

23.1.3 Funktionen

23.1.3.1 Zuordnung von Dienst und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Rühren EXTENDS ServiceControl  
VAR  
    Parameter1: StringConfigParam := (TagName := 'Parameter1', TagDescription := '');  
END_VAR  
  
(* Beispielimplementierung ST *)  
SUPER^();  
Parameter1(Service := THIS^);
```

23.1.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden.

23.1.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert als „VOut“ als Ausgang am FB ausgegeben. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst vom Zustand Offline in den Zustand Operator oder Automatic wechselt – ansonsten bleibt VOut konstant.

23.1.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

23.1.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

23.1.4 Source Mode

23.1.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

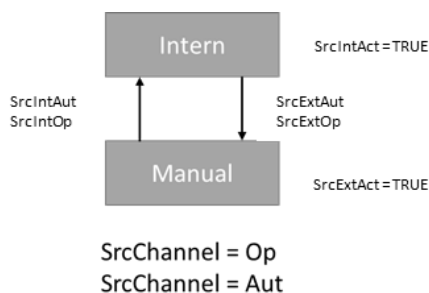


Fig. 155 Zustandsdiagramm Source Mode

23.1.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

23.1.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

23.1.5 Operation Mode

23.1.5.1 Kurzbeschreibung

Der Operation Mode ist einen Zustandsautomaten mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

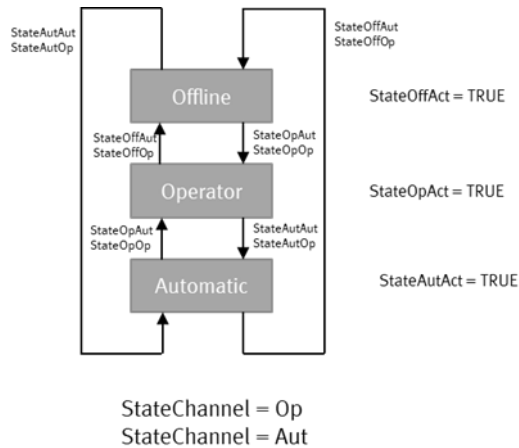


Fig. 156 Zustandsdiagramm Operation Mode

23.1.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

23.1.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.

- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

23.1.5.4 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter uneindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

23.2 StringProcedureParam

23.2.1 Kurzbeschreibung

Der Funktionsbaustein (FB) StringProcedureParam kann zur Definition eines String-basierte Prozedurparameters (Zeichenkette) für eine Dienst-Prozedur genutzt werden. Der aktuelle Wert des Parameters liegt am Ausgang VOut an. Gegenüber einer einfachen Wertvorgabe bspw. unter Nutzung einer VAR_INPUT vom Typ STRING bietet der Typ folgende Erweiterungen:

- Der Baustein verfügt analog zu einem Dienst-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern).
- Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant. Dadurch eignet sich ein Prozedurparameter vor allem zur Vorgabe von Werten, die für eine bestimmte Prozedur eines Dienstes notwendig sind und sich während der Ausführung der Prozedur nicht ändern dürfen (bspw. die Vorgabe einer Zeit für eine zeitbeschränkte Dosieren-Prozedur).

23.2.2 Grundfunktionen

23.2.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

23.2.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 146 Worst Quality Code Enumeration

23.2.2.3 OSLevel

Die Variable OSLevel ermöglicht einen einfachen Mechanismus, mit dem das Schreiben mehrerer Bedienungsebenen überwacht werden kann. Jede Bedienungsebene (z.B. lokaler Leitstand / zentraler Leitstand) wird einer entsprechenden Zahl zugeordnet. Wenn die Zahl der Bedienungsebene und die Zahl dieser Variable übereinstimmen, sollen die Bedienelemente der Bedienungseinrichtung den Schreibbefehl zulassen. Andernfalls sollen die Eingabefelder inaktiv sein. Somit werden gleichzeitige Schreibzugriffe von mehreren Bedienungsebenen verhindert.

Die Variable löst im FB keinerlei Funktionalität aus und wird auch nicht geprüft. Sie dient lediglich als persistenten Ort der Speicherung, damit alle Bedienstationen auf ein und denselben Wert zugreifen können. Die Variable steht nicht als Eingang am FB zur Verfügung, sondern kann nur über OPC UA beschrieben werden.

23.2.3 Funktionen

23.2.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

(* Beispieldeklaration *)

```
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
```

```
VAR
```

```
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
    ProcedureID := 1, IsSelfCompleting := TRUE);
```

```
    Zeit: StringProcedureParam := (TagName := 'Zeit', TagDescription := '', ProcedureID := 1);  
END_VAR
```

(* Beispielimplementierung ST *)

```
SUPER^();
```

```
Zeitbeschränkt(Service := THIS^);
```

```
Zeit(Service := THIS^);
```

23.2.3.2 Eingangswert (Operator/Intern/Extern)

Da es drei Eingangswerte gibt, muss entschieden werden welcher der Werte aktiv ist (VOp, VInt oder VExt) und für die Berechnung des Ausgangs VOut weiterverarbeitet wird. Dazu steht der Source Mode zur Verfügung, welcher die beiden Eingänge steuert. Bei der Initialisierung ist die manuelle Quelle und somit VOp aktiv. Die Funktionsweise und Handhabung des Source Modes wird in einem extra Abschnitt erläutert. VOp und VExt sind keine Eingänge am FB, sondern können nur als OPC UA Variable gesetzt werden.

23.2.3.3 Ausgangswert „VOut“

Der gewählte Eingangswert als „VOut“ als Ausgang am FB ausgegeben. Eine Aktualisierung von VOut erfolgt jeweils nur zu dem Zeitpunkt, wenn der dem Parameter zugeordnete Dienst mit der dem Parameter zugeordneten Prozedur gestartet wird (d.h. den Zustand Starting durchläuft) – ansonsten bleibt VOut konstant.

23.2.3.4 Ausgang „VRbk“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

23.2.3.5 Ausgang „VReq“

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch auf die Normierungswerte begrenzt. „VReq“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

23.2.4 Source Mode

23.2.4.1 Kurzbeschreibung

Der Source Mode ist ein Zustandsautomat zur Auswahl einer Quelle, wenn ein Wert von zwei unterschiedlichen Quellen beeinflusst werden kann.

Es wird hierbei unterschieden zwischen interner und manueller Wertevorgabe. Über den Eingang „SrcChannel“ am FB wird festgelegt, ob direkt der FB den Source Mode umschalten (intern) darf oder ob ein Operator (external) über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der SrcChannel-Variable durch den FB definiert werden.

Ist die interne Quelle aktiv (VInt) so wird der Ausgang „SrcIntAct“ auf TRUE gesetzt. Ist die manuelle Quelle (VExt) aktiv, so wird das Ausgangsbit „SrcExtAct“ auf TRUE gesetzt.

Standardmäßig ist die interne Quelle aktiv.

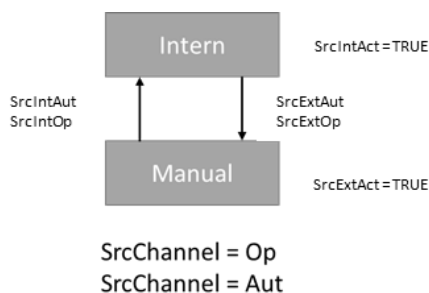


Fig. 157 Zustandsdiagramm Source Mode

23.2.4.2 Zugriff modul-intern (direkt am FB)

Um den Source Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „SrcIntAut“ (umschalten von Manual zu Intern) oder „SrcExtAut“ (umschalten von Manual zu Intern) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so wird „SrcIntAut“ und „SrcExtAut“ auf FALSE gesetzt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntAut“ und „SrcExtAut“) gleichzeitig gesetzt, so hat „SrcIntAut“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

23.2.4.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „SrcIntOp“ und „SrcExtOp“.

Damit ein Wechsel der Source überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „SrcChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „SrcIntOp“ (internal Source) oder „SrcExtOp“ (externe Source) auf TRUE setzen.
- Nach der Anfrage werden beide Variablen („SrcIntOp“ und „SrcExtOp“) vom FB auf FALSE gesetzt – d.h. Anfrage wurde vom FB erkannt.
- Über die Ausgänge „SrcIntAct“ und „SrcExtAct“ kann die aktuell eingestellte Quelle abgelesen werden. Je nachdem welche Quelle aktiv ist, wird der entsprechende Ausgang auf TRUE gesetzt.

Setzt der FB die Eingänge wieder auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „SrcChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden beide Eingänge („SrcIntOp“ und „SrcExtOp“) gleichzeitig gesetzt, so hat „SrcIntOp“ die höhere Priorität und die Quelle wechselt bzw. bleibt auf Intern.

Hinweis:

Der Source Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „SrcIntOp“, „SrcExtOp“, „SrcIntAut“ oder „SrcExtAut“ an den FB geschickt, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

23.2.5 Operation Mode

23.2.5.1 Kurzbeschreibung

Der Operation Mode ist ein Zustandsautomat mit drei Zuständen, der festlegt, ob sich der FB im Offline, Operator oder Automatic Mode befindet. Über den Eingang „StateChannel“ am FB wird festgelegt, ob direkt der FB den Operation Mode umschalten darf oder ob ein Operator über ein Faceplate Änderungen vornehmen darf. In Abhängigkeit dieser Variable sind entweder die Operator-Signale (*Op-Variablen) oder die internen Eingangsvariablen am FB (*Aut-Variablen) ausschlaggebend. Der Zeitpunkt, in dem umgeschaltet werden darf kann mit Hilfe der StateChannel-Variable durch den FB definiert werden. Als Initialisierungs-State befindet sich der FB immer in Offline.

Ist der Offline Mode aktiv so wird der Ausgang „StateOffAct“ auf TRUE gesetzt. Ist der Operator Mode aktiv so wird „StateOpAct“ und beim Automatic Mode „StateAutAct“ auf TRUE gesetzt.

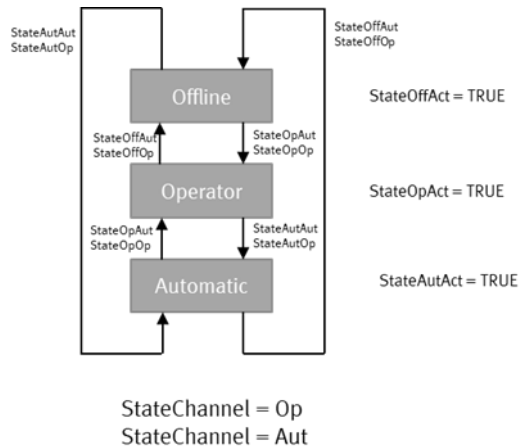


Fig. 158 Zustandsdiagramm Operation Mode

23.2.5.2 Zugriff modul-intern (direkt am FB)

Um den Operation Mode direkt am FB umzustellen, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Aut“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- Eingang „StateOffAut“ (umschalten in Offline Mode), „StateOpAut“ (umschalten in Operator Mode) oder „StateAutAut“ (umschalten in Automatic Mode) auf TRUE setzen.
- Wurde die Anfrage vom FB zur Kenntnis genommen so werden die Eingänge wieder auf FALSE gesetzt.
- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffAut“/„StateOpAut“/„StateAutAut“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und die Quelle wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, so hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

23.2.5.3 Zugriff modul-extern (OPC UA)

Der Zugriff über OPC UA erfolgt über die Variablen „StateOffOp“, „StateOpOp“ und „StateAutOp“.

Damit ein Wechsel des Zustandes überhaupt über OPC UA angefragt werden kann, muss wie folgt vorgegangen werden:

- „StateChannel“ muss über die Enumeration ChannelAutOp auf „Op“ gesetzt sein. (Kann nur am FB oder über das Faceplate des PA Toolkits umgestellt werden)
- „StateOffOp“ (Offline), „StateOpOp“ (Operator) oder „StateAutOp“ (Automatic) auf TRUE setzen.
- Nach der Anfrage müssen alle drei Variablen („StateOffOp“, „StateOpOp“ oder „StateAutOp“) vom FB auf FALSE gesetzt werden – d.h. die Anfrage wurde vom FB erkannt.

- Über die Ausgänge „StateOffAct“, „StateOpAct“ oder „StateAutAct“ kann der aktive Mode abgelesen werden. Das Ausgangsbit des aktiven Modes wird auf TRUE gesetzt.

Setzt der FB die Eingänge auf FALSE heißt es nicht automatisch, dass der Befehl ausgeführt wird, es heißt nur, dass dieser vom FB empfangen wurde. Ist z.B. „StateChannel“ auf Op-Variablen gesetzt, so werden Anfragen vom FB nicht verarbeitet, jedoch zur Kenntnis genommen. Ungültige Anfragen werden nicht für einen späteren Zeitpunkt gespeichert.

Werden zwei oder mehr Eingänge („StateOffOp“/„StateOpOp“/„StateAutOp“) gleichzeitig gesetzt, so hat der Offline Mode die höhere Priorität und der Zustand wechselt bzw. bleibt im Offline Mode. Da vom Automatic Mode nicht in den Offline Mode direkt gewechselt werden kann, hat hier der Operator Mode eine höhere Priorität als der Automatic Mode.

Hinweis:

Der Operation Mode arbeitet nach einem Handshake-Verfahren. Wird ein Steuerbefehl über die Variablen „StateOffOp“, „StateOpOp“, „StateAutOp“, „StateOffAut“, „StateOpAut“, „StateAutAut“, wird diese Anfrage bestätigt, in dem die Variable wieder auf FALSE“ gesetzt wird. Aus diesem Grund ist es wichtig diese Eingänge am FB nicht fest zu belegen.

23.2.5.4 Synchronisierung von Operation und Source Mode

Wie oben beschrieben verfügt der Parameter-Baustein über einen eigenen Zustand (Offline/Operator/Automatic) und ermöglicht Wertevorgaben aus unterschiedlichen Quellen (Operator/Intern/Extern). Da der Parameter uneindeutig einem bestimmten Dienst zugeordnet ist, ist es in vielen Fällen sinnvoll, sowohl den Zustand als auch den Kanal für die Wertvorgabe zwischen Dienst und Parameter zu synchronisieren. Hierzu kann der Eingang Sync genutzt werden. Ist dieser auf True gesetzt (Default-Wert), wird der Parameter-eigene Operation Mode und Source Mode (s. unten) ignoriert und stattdessen der aktuelle Operation Mode und Source Mode des zugeordneten Dienstes genutzt.

Hinweis:

Geht der zugeordnete Dienst bei aktivierter Synchronisierung in den Offline-Zustand, verbleibt der Parameter im letzten Zustand. Soll der Parameter ebenfalls in Offline gehen, muss diese Überführung separat herbeigeführt werden.

23.3 StringProcessValueOut

23.3.1 Kurzbeschreibung

Der Funktionsbaustein (FB) StringProcessValueOut dient zum kontinuierlichen Rückmelden eines String-basierten Wertes (Zeichenkette) aus einer Dienst-Prozedur.

23.3.2 Grundfunktionen

23.3.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

23.3.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 147 Worst Quality Code Enumeration

23.3.3 Funktionen

23.3.3.1 Zuordnung von Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist.

```
(* Beispieldeklaration *)  
FUNCTION_BLOCK Rühren EXTENDS ServiceControl  
VAR  
    Zeitbeschaenkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',  
ProcedureID := 1, IsSelfCompleting := TRUE);  
    Fuellstand: StringProcessValueOut := (TagName := 'Fuellstand', TagDescription := '', ProcedureID :=  
1);  
END_VAR  
  
(* Beispielimplementierung ST *)  
SUPER^();  
Zeitbeschränkt(Service := THIS^);  
Fuellstand();
```

23.3.3.2 Eingangswert „Text“

Über die Eingangsvariable „Text“ kann eine Zeichenfolge hinterlegt werden. Die Zeichenfolge kann direkt eingegeben oder über eine Variable hinterlegt werden.

23.4 StringReportValue

23.4.1 Kurzbeschreibung

Der Funktionsbaustein (FB) StringReportValue dient zur Bereitstellung eines String-basierten Wertes (Zeichenkette) aus einer Dienst-Prozedur zu Dokumentations- bzw. Nachweiszwecken. Dieser FB erweitert die Funktionalität eines StringProcessValueOut derart, dass der Wert des Parameters („V“) eingefroren wird, wenn sich der zugehörige Dienst in einem der Zustände Completed, Stopped oder Aborted befindet. Dies ermöglicht es einem angeschlossenen Dokumentations-System, den Wert des Parameters in diesen Zuständen bspw. für Nachweiszwecke auszulesen.

23.4.2 Grundfunktionen

23.4.2.1 Identifizierung

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

23.4.2.2 Worst Quality Code (WQC)

Der Worst Quality Code gibt den aktuellen Zustand des FBs und dessen Signale an. Der schlechteste Zustand hat immer die höchste Priorität und wird am Ausgang „WQC“ ausgegeben. In der folgenden Tabelle sind alle Beschreibungen aufgelistet. Eine spezifische Fehlerbehandlung für diesen FB, wird im Abschnitt FB-spezifischer Worst Quality Code, erläutert.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 148 Worst Quality Code Enumeration

23.4.3 Funktionen

23.4.3.1 Zuordnung Dienst, Prozedur und Parameter

Es besteht grundsätzlich eine eindeutige Zuordnung zwischen Dienst, Prozedur und Parameter. Um diese Zuordnung deutlich zu machen, ist ein Parameter grundsätzlich im Deklarationsteil des entsprechenden Service-FBs zu deklarieren und auch zu initialisieren. Bei der Initialisierung ist darüber hinaus über den Parameter ProcedureID die Prozedur zu definieren, der der Parameter zugeordnet ist. Zusätzlich ist dem Parameter-FB beim Aufruf der zugeordnete Dienst über den Eingang Service mitzuteilen.

```
(* Beispieldeklaration *)
FUNCTION_BLOCK Rühren EXTENDS ServiceControl
VAR
    Zeitbeschränkt: ServiceProcedure := (TagName := 'Zeitbeschränkt', TagDescription := '',
    ProcedureID := 1, IsSelfCompleting := TRUE);
    Fuellstand: AnaReportValue := (TagName := 'Fuellstand', TagDescription := '', ProcedureID := 1);
END_VAR

(* Beispielimplementierung ST *)
SUPER^();
Zeitbeschränkt(Service := THIS^);
Fuellstand(Service := THIS^);
```

23.4.3.2 Eingangswert „TextIn“

Über die Eingangsvariable „TextIn“ kann eine Zeichenfolge hinterlegt werden. Die Zeichenfolge kann direkt eingegeben oder über eine Variable hinterlegt werden.

24 Parameterbausteine

Damit Service-Parameter in der Visualisierung dargestellt und gesetzt werden können, gibt es drei Parameter-Funktionsbausteine. Es gibt ein für REAL, STRING- und BOOL-Parameter. Jeder FB hat Eingänge, mit denen die Service-Parameter verbunden werden können. In der Visualisierung muss das entsprechende Parameter-Symbol zum Service hinzugefügt werden. Während der Laufzeit kann per Klick auf das Parameter-Symbol das Faceplate mit den verknüpften Parametern geöffnet werden. Hier können Parameter geschrieben und gelesen werden. Die FBs haben keine weiteren Funktionen sie dienen nur für die Darstellung in der Visualisierung.

24.1 Symbole

24.1.1.1 hBoolParameter

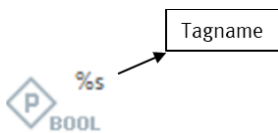


Fig. 159 Symbol hBoolParameter

24.1.1.2 hRealParameter

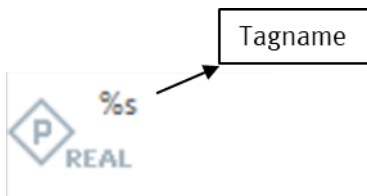


Fig. 160 Symbol hRealParameter

24.1.1.3 hStringParameter



Fig. 161 Symbol hStringParameter

24.2 Faceplate Bool-Parameter

24.2.1.1 Param 1-4

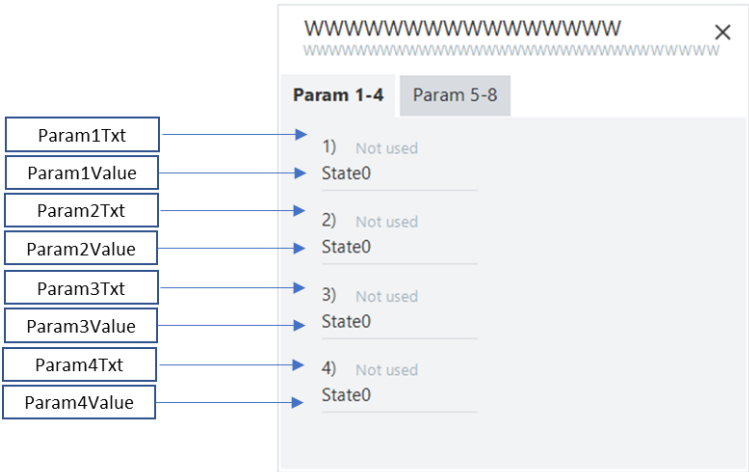


Fig. 162 Bool-Parameter Param1-4 Faceplate

24.2.1.2 Param 5-8

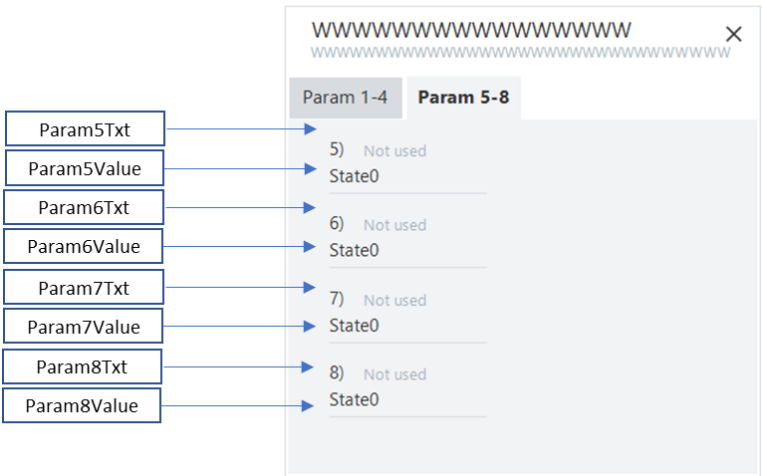


Fig. 163 Bool-Parameter Param5-8 Faceplate

24.3 Faceplate Real-Parameter

24.3.1.1 Param 1-4

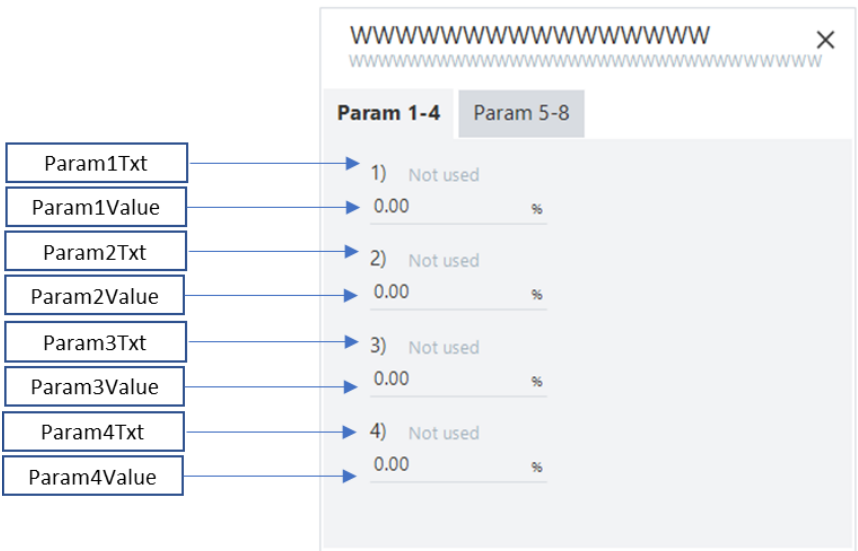


Fig. 164 Real-Parameter Param1-4 Faceplate

24.3.1.2 Param 5-8

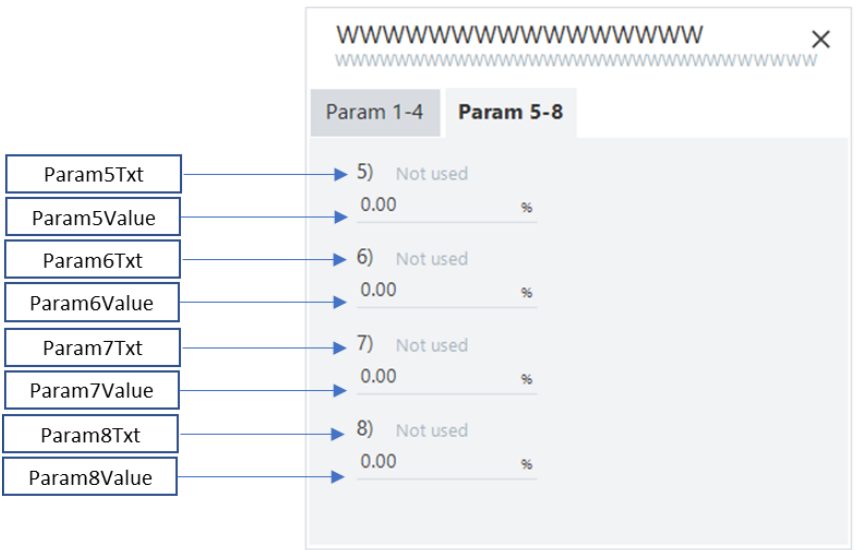


Fig. 165 Real-Parameter Param5-8 Faceplate

24.3.2 Faceplate String-Parameter

24.3.2.1 Param 1-4

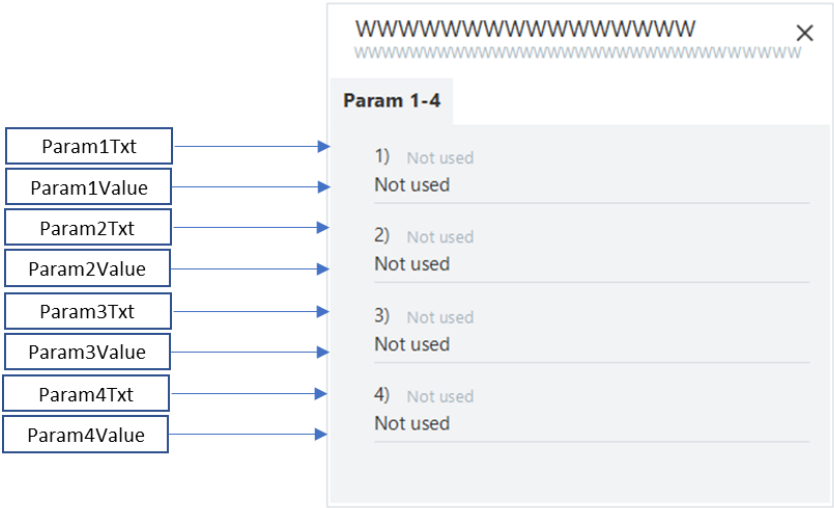


Fig. 166 String-Parameter Param1-4 Faceplate

25 Hilfsfunktionen & Enumeration

25.1 Hilfsfunktionen

25.1.1 Scaler

Der Funktionsbaustein Scaler kann für die Umrechnung von einem Eingangswert „FV“ in einen Ausgangswert „V“ verwendet werden. Dazu werden zusätzlich die Skalierungswerte von „FV“ („FVNormMin“ und „FVNormMax“) und dem Ausgang („VNormMin“ und „VNormMax“) übergeben. Als Ausgangswert wird ein WQC (Worst Quality Code) zurückgegeben, sowie der umgerechnete Wert „V“ als REAL und als UINT.

Folgende WQCs können bei diesem FB auftreten:

FB-spezifischer Worst Quality Code				
Byte	Byte	Priorität	Beschreibung	Grund/Fehler
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
128d	16#80	6	Good	Kein Fehler
255d	16#FF	7	No QC available	Wert bei jedem Zyklusbeginn

Tab. 149 Scaler spezifischer Worst Quality Code

25.2 Enumeration

25.2.1 Enumeration WorstQualityCode

WorstQualityCode ist eine Enumeration und stellt die verschiedenen Statusanzeigen welcher der Worst Quality Code annehmen kann, dar.

Worst Quality Code Enumeration

Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 150 Worst Quality Code Enumeration

25.2.2 Enumeration UnitSettings

In der Enumeration sind alle Einheiten hinterlegt, die im PA Toolkit verwendet werden können.

25.2.3 Enumeration ChannelAutOp

Die Enumeration ist zum Umschalten der Op und Aut-Variablen an einem FB vorgesehen. Jeder FB der einen Source Mode verwendet hat einen Eingang „SrcChannel“. Wird der Operation Mode verwendet hat jeder FB einen Eingang „StateChannel“. An diesem kann eingestellt werden, ob zur Steuerung die OP-Variablen oder die Aut-Variablen verwendet werden sollen. Die Enumeration besteht aus dem Eintrag „Aut“ und „Op“ und kann genau für diese zwei Eingänge am FB verwendet werden.

25.2.4 Enumeration LockLogic

Bei der Verriegelungsanzeige kann die logische Grundoperation eingestellt werden. Dieses Enumeration liefert die zwei logischen Möglichkeiten: AND und OR.

25.2.5 Enumeration Command

Command ist eine Enumeration, welche die verschiedenen möglichen Kommandos zum Auslösen von Zustandswechseln der Zustandsmaschine eines Dienst-Bausteins (ServiceControl, SimpleContinuousServiceControl und SimpleSelfCompletingServiceControl) darstellt.

Die Werte sind dabei gemäß folgender Tabelle codiert:

Wert (dezimal)	Wert (hexadezimal)	Name
0	16#0	NoCommand
2	16#2	Reset
4	16#4	Start
8	16#8	Stop
16	16#10	Hold
32	16#20	Unhold
64	16#40	Pause
128	16#80	Resume
256	16#100	Abort
512	16#200	Restart
1024	16#400	Complete

Tab. 151 Command Enumeration

25.2.6 Enumeration SMState

SMState ist eine Enumeration, welche die verschiedenen möglichen Zustände der Zustandsmaschine eines Dienst-Bausteins (ServiceControl, SimpleContinuousServiceControl und SimpleSelfCompletingServiceControl) darstellt.

Die Werte sind dabei gemäß folgender Tabelle codiert:

Wert (dezimal)	Wert (hexadezimal)	Name
4	16#4	Stopped
8	16#8	Starting
16	16#10	Idle
32	16#20	Paused
64	16#40	Execute
128	16#80	Stopping
256	16#100	Aborting
512	16#200	Aborted
1024	16#400	Holding
2048	16#800	Held
4096	16#1000	Unholding
8192	16#2000	Pausing
16384	16#4000	Resuming
32768	16#8000	Resetting
65536	16#10000	Completing
131072	16#20000	Completed

Tab. 152 SMState Enumeration